Made available by Hasselt University Library in https://documentserver.uhasselt.be

A framework for comparing query languages in their ability to express boolean queries

Peer-reviewed author version

SURINX, Dimitri; VAN DEN BUSSCHE, Jan & Van Gucht, Dirk (2019) A framework for comparing query languages in their ability to express boolean queries. In: ANNALS OF MATHEMATICS AND ARTIFICIAL INTELLIGENCE, 87(1-2), p. 157-184.

DOI: 10.1007/s10472-019-09639-5
Handle: http://hdl.handle.net/1942/29982

Noname manuscript No.
(will be inserted by the editor)

A framework for comparing query languages in their ability to express boolean queries*

Dimitri Surinx · Jan Van den Bussche · Dirk Van Gucht

Keywords Query Languages · Expressive Power · Graph Databases · Conjunctive Queries · Boolean Query Modalities

Abstract For any query language $\mathcal{F}$, we consider three natural families of boolean queries. Nonemptiness queries are expressed as $e \neq \emptyset$ with e an $\mathcal{F}$ expression. Emptiness queries are expressed as $e = \emptyset$. Containment queries are expressed as $e_1 \subseteq e_2$. We refer to syntactic constructions of boolean queries as modalities. In first order logic, the emptiness, nonemptiness and containment modalities have exactly the same expressive power. For other classes of queries, e.g., expressed in weaker query languages, the modalities may differ in expressiveness. We propose a framework for studying the expressive power of boolean query modalities. Along one dimension, one may work within a fixed query language and compare the three modalities. Here, we identify crucial query features that enable us to go from one modality to another. Furthermore, we identify semantical properties that reflect the lack of these query features to establish separations. Along a second dimension, one may fix a modality and compare different query languages. This second dimension is the one that has already received quite some attention in the literature, whereas in this paper we emphasize the first dimension. Combining both dimensions, it is interesting to compare the expressive power of a weak query language using a strong modality, against that of a seemingly stronger query language but perhaps using a weaker modality. We present some initial results within this theme. The two main query languages to which we apply our framework are the algebra of binary relations, and the language of conjunctive queries.

* This is a post-peer-review, pre-copyedit version of an article accepted for publication in Annals of Mathematics and Artificial Intelligence. The final authenticated version still has to appear.

D. Surinx (corresponding author) · J. Van den Bussche
Hasselt University, School for Information Technology
Martelarenlaan 42, 3500 Hasselt, Belgium
E-mail: {dimitri.surinx,jan.vandenbussche}@uhasselt.be

D. Van Gucht
Indiana University, School of Informatics, Computing, and Engineering
Luddy Hall, 700 N Woodlawn Ave, Bloomington, Indiana 47408, United States

1 Introduction

When a relational database is queried, the result is normally a relation. Some queries, however, only require a yes/no answer; such queries are often called *boolean queries*. We may ask, for example, "is student 14753 enrolled in course c209?" Also, every integrity constraint is essentially a boolean query. Another application of boolean queries is given by SQL conditions, as used in updates and triggers, or in if-then-else statements of SQL/PSM (PL/SQL) programs.

In the theory of database query languages and in finite model theory [1,13, 21,20], it is standard practice to express boolean queries under what we call the *nonemptiness modality*. Under this modality, boolean queries are expressed in the form $e \neq \emptyset$ where e is a query expression in some query language. For example, under the nonemptiness modality, the above boolean query "is student 14753 enrolled in course c209?" is expressed by the nonemptiness of the query "give all students with id 14753 that are enrolled in course c209". The nonemptiness modality is used in practice in the query language SPARQL. In that language, the result of a boolean query `ASK` P is true if and only if the corresponding query `SELECT *` P has a nonempty result. Another example of the nonemptiness modality in practice is given by SQL conditions of the form `EXISTS (`Q`)`.

Sometimes, however, an integrity constraint is more naturally expressed by a query that looks for violations; then the constraint holds if the query returns no answers. So, here we use the *emptiness* modality rather than nonemptiness. For example, to express the integrity constraint that students can be enrolled in at most ten courses, we write a query retrieving all students enrolled in more than ten courses. The query must return an empty result; otherwise an error is raised. SQL conditions of the form `NOT EXISTS (`Q`)`, instrumental in formulating nonmonotone queries, obviously use the emptiness modality.

Yet another natural modality is *containment* of the form $e_1 \subseteq e_2$, where e_1 and e_2 are two query expressions. This boolean query returns true on a database D if $e_1(D)$ is a subset of $e_2(D)$.[1] For example, the integrity constraint "every student taking course c209 should have passed course c106" is naturally expressed by $e_1 \subseteq e_2$, where e_1 is the query retrieving all students taking c209 and e_2 is the query retrieving all students that passed c106. An embedded tuple-generating dependency [1,8] can be regarded as the containment of two conjunctive queries. Similarly, an equality-generating dependency [1,8] can be regarded as the containment of a conjunctive query in the query returning all identical pairs of data elements.

This brings us to the main motivation of this paper: by using the containment modality, one can use a weaker query language, such as conjunctive queries, and still be able to express integrity constraints that would not be expressible in the language using, say, the nonemptiness modality. A weaker language is easier to use and queries can be executed more efficiently. We find it an intriguing question how different modalities compare to each other, under different circumstances de-

[1] In this paper, $e_1 \subseteq e_2$ stands for a boolean query which, in general, may return true on some databases and return false on the other databases. Thus $e_1 \subseteq e_2$ as considered in this paper should not be misconstrued as an instance of the famous query containment problem [11, 1], where the task would be to verify statically if $e_1(D)$ is a subset of $e_2(D)$ on *every* database D. Indeed, if e_1 is contained in e_2 in this latter sense, then the boolean query $e_1 \subseteq e_2$ is trivial as it returns true on every database.

pending on the query language at hand. Furthermore, one may want to compare the expressiveness of different query languages across different modalities for expressing boolean queries. Moreover, we can observe that the emptiness modality is simply the negation of the nonemptiness modality. Inspired by this, we are interested in understanding under what circumstances the containment modality is closed under negation, or under other boolean connectives such as conjunction.

We can illustrate the above questions using well-known simple examples.

Example 1 A referential integrity constraint (inclusion dependency) is clearly expressible by a containment of conjunctive queries (CQs), but not by the nonemptiness of a CQ. This is simply because CQs are monotone, whereas an inclusion dependency is not. On the other hand it is neither expressible by the emptiness of a CQ, because such boolean queries are antimonotone whereas again inclusion dependencies are not.

For another example, a key constraint (functional dependency, FD) is again not monotone, so again not expressible by the nonemptiness of a monotone query. An FD is, however, naturally expressed by the *emptiness* of a CQ with nonequalities. For example, a relation $R(A, B, C)$ satisfies the FD $A \to B$ if and only if the result of

$$() \leftarrow R(x, y_1, z_1), R(x, y_2, z_2), y_1 \neq y_2$$

is empty. An FD is also expressible as a containment of two CQs. For example, the above FD holds if and only if the containment

$$(x, y_1, y_2) \leftarrow R(x, y_1, z_1), R(x, y_2, z_2) \quad \subseteq \quad (x, y, y) \leftarrow R(x, y, z)$$

is satisfied. □

In this paper, we attack the above questions from several angles. We begin by comparing the three basic modalities: emptiness, nonemptiness, containment, in the general context of an arbitrary query language. In such a context it is possible to formulate sufficient conditions for, say, emptiness queries to be convertible to nonemptiness queries, or nonemptiness queries to be convertible to containment queries. For example, if we have a sufficiently powerful query language that can express cylindrification and complementation, such as full first-order logic, then it does not really matter which boolean query modality one uses. Conversely, we also formulate some general properties of query languages, like monotonicity or additivity, that preclude the conversion of one modality into another.

Our second angle is to consider a range of specific query languages and characterize how the different modalities compare to each other, for each language in this range. It would be very natural to do this for Codd's relational algebra, where we obtain a range of fragments by varying the allowed operations. For example, we may allow attribute renaming or not, or we may allow set difference or not. While such an investigation remains to be done, in this paper, we have opted to work with the algebra of *binary relations*. This algebra can be seen as a more controlled setting of the relational algebra, and also serves as a well-established and adopted formalization of graph query languages [4, 30, 6, 23, 10, 14, 22, 3]. We will completely characterize how the different boolean query modalities compare for each fragment of the algebra of binary relations. Apart from this algebra, we will also look at the popular class of conjunctive queries under the light of the three boolean query modalities.

Our third angle is to investigate how the expressiveness of two different query languages can be compared when using a different modality for each language. One can, for example, compare a stronger language under the weak emptiness modality, to a weak language under the stronger containment modality. The FD example in Example 1 clearly follows this pattern. In this theme we use earlier and new results [29,26] to show separations between the nonemptiness modality and the containment modality for different fragments of the algebra of binary relations. There remain some open problems in this theme, which we will summarize.

Finally, we look at the question of when a class of boolean queries is closed under conjunction, or under negation. Especially the question of closure under conjunction for boolean queries expressed as containments is very interesting with some open problems remaining.

This article extends the conference paper [28] by providing full proofs, many of which were missing from the conference paper.

Previous work. In our previous work we have compared the expressive power of fragments of the algebra of binary relations under the nonemptiness modality [16, 15,29] and under the containment modality [27]. The present paper is complementary in that it emphasizes the comparison of different boolean query modalities, for fixed fragments or across fragments. Moreover, this paper also deals with general query languages over general relational databases, and also with the language of conjunctive queries.

2 Preliminaries

A database schema S is a finite nonempty set of relation names. Every relation name R is assigned an arity, which is a natural number. Assuming some fixed infinite universe of data elements V, an instance I of a relation name R of arity k is a finite k-ary relation on V, i.e., a subset of $V^k = V \times \cdots \times V$ (k times). More generally, an instance I of a database schema S assigns to each $R \in S$ an instance of R, denoted by $I(R)$. The active domain of an instance I, denoted by $\mathrm{adom}(I)$, is the set of all data elements from V that occur in I. For technical reasons we exclude the *empty instance*, i.e., one of the relations in I must be nonempty. Thus, $\mathrm{adom}(I)$ is never empty.

For a natural number k, a k-ary query over a database schema S is a function that maps each instance I of S to a k-ary relation on $\mathrm{adom}(I)$. We require queries to be generic [1]. A query q is generic if for any permutation f of the universe V, and any instance I, we have $q(f(I)) = f(q(I))$.

We assume familiarity with the standard relational query languages such as first-order logic and relational algebra [1].

2.1 Tests, Cylindrification, Complementation.

Let q_1 and q_2 be queries over a common database schema. We define the query $(q_1 \text{ if } q_2)$ as follows:

$$(q_1 \text{ if } q_2)(I) = \begin{cases} q_1(I) & \text{if } q_2(I) \neq \emptyset; \\ \emptyset & \text{otherwise.} \end{cases}$$

Naturally, we say that a family $\mathcal{F}$ of queries over a common database schema is *closed under tests* if for any two queries q_1 and q_2 in $\mathcal{F}$, the query $(q_1 \,\mathrm{if}\, q_2)$ is also in $\mathcal{F}$.

Cylindrification is an operation on relations that, like projection, corresponds to existential quantification, but, unlike projection, does not reduce the arity of the relation [19,9]. We introduce an abstraction of this operation as follows. For any natural number k and query q, we define the *k-ary cylindrification of q*, denoted by $\gamma_k(q)$, as follows:

$$\gamma_k(q)(I) = \begin{cases} \mathrm{adom}(I)^k & \text{if } q(I) \neq \emptyset; \\ \emptyset & \text{otherwise.} \end{cases}$$

This definition works for any query q; the arity of q need not be k. We say that a family $\mathcal{F}$ of queries over a common database schema is *closed under k-ary cylindrification* ($k \geq 1$) if for any query $q \in \mathcal{F}$, the query $\gamma_k(q)$ is also in $\mathcal{F}$.

Example 2 Let S be a schema with two ternary relations R and T, and let q be the 3-ary query that maps any instance I over S to $I(R) - I(T)$. Then, $\gamma_1(q)$ is the unary query that maps any instance I over S to $\mathrm{adom}(I)$ if $I(R) \not\subseteq I(T)$ and to $\emptyset$ otherwise. □

For a k-ary query q, the *complement* of q, denoted by q^c, is defined by $q^c(I) = \mathrm{adom}(I)^k - q(I)$ (set difference). We say that a family $\mathcal{F}$ of queries over a common database schema is *closed under k-complementation* if for any query $q \in \mathcal{F}$ of arity k, the query q^c is also in $\mathcal{F}$.

Finally, we say that a family $\mathcal{F}$ of queries over a common database schema is *closed under set difference* if for any two queries $q_1, q_2 \in \mathcal{F}$ of the same arity, the query q that maps instances I onto $q_1(I) - q_2(I)$ is also in $\mathcal{F}$.

2.2 Navigational graph query languages.

Some of the general discussion, framework and results apply to general query languages over general relational databases, or to the language of conjunctive queries over general relational databases. On the other hand, many technical results concern graph databases, corresponding with the case where the database schema S is restricted to only binary relation names. Any instance I of S can be considered as a graph G, where the elements of the active domain are considered as nodes, the pairs in the binary relations are directed edges, and the relation names are edge labels.

Navigational queries on graph databases are formalized as binary queries on graphs. The most basic navigational language we consider is the algebra $\mathcal{N}_S$. The expressions of this algebra are built recursively from the relation names in S and the primitives $\emptyset$ and id, using the operators composition $(e_1 \circ e_2)$ and union $(e_1 \cup e_2)$. Semantically, each expression denotes a query in the following way.

$$\begin{aligned} \mathsf{id}(G) &= \{(m,m) \mid m \in \mathrm{adom}(G)\} \\ R(G) &= G(R) \qquad \text{for relation name } R \in S \\ \emptyset(G) &= \emptyset \\ e_1 \circ e_2(G) &= \{(m,n) \mid \exists p : (m,p) \in e_1(G) \wedge (p,n) \in e_2(G)\} \\ e_1 \cup e_2(G) &= e_1(G) \cup e_2(G). \end{aligned}$$

Although the assumption of a basic language is a point of discussion, it can be argued that our choice of basic language is not unreasonable [27].

The basic algebra $\mathcal{N}_S$ can be extended by adding some of the following features: the primitives diversity (di), and the full relation (all); and the operators converse (e^{-1}), intersection ($e_1 \cap e_2$), set difference ($e_1 - e_2$), projections ($\pi_1(e)$ and $\pi_2(e)$), coprojections ($\overline{\pi}_1(e)$ and $\overline{\pi}_2(e)$), and transitive closure (e^+). We refer to the operators in the basic algebra as *basic features*; we refer to the extensions as *nonbasic features*. The semantics of the extensions are as follows:

$$\begin{aligned}
\mathsf{di}(G) &= \{(m,n) \mid m,n \in \mathrm{adom}(G) \wedge m \neq n\} \\
\mathsf{all}(G) &= \{(m,n) \mid m,n \in \mathrm{adom}(G)\} \\
e^{-1}(G) &= \{(m,n) \mid (n,m) \in e(G)\} \\
e_1 \cap e_2(G) &= e_1(G) \cap e_2(G) \\
e_1 - e_2(G) &= e_1(G) - e_2(G) \\
\pi_1(e)(G) &= \{(m,m) \mid m \in \mathrm{adom}(G) \wedge \exists n : (m,n) \in e(G)\} \\
\pi_2(e)(G) &= \{(m,m) \mid m \in \mathrm{adom}(G) \wedge \exists n : (n,m) \in e(G)\} \\
\overline{\pi}_1(e)(G) &= \{(m,m) \mid m \in \mathrm{adom}(G) \wedge \neg\exists n : (m,n) \in e(G)\} \\
\overline{\pi}_2(e)(G) &= \{(m,m) \mid m \in \mathrm{adom}(G) \wedge \neg\exists n : (n,m) \in e(G)\} \\
e^+(G) &= \text{the transitive closure of } e(G).
\end{aligned}$$

All the above operators are well-established in so-called "navigational" graph querying [23, 10, 14, 22, 3].

A *fragment* is any set F of nonbasic features, in which we either take both projections π_1 and π_2 or none of them, and the same for coprojection.[2]

If F is a fragment, we denote by $\mathcal{N}_S(F)$ the language obtained by adding the features in F to $\mathcal{N}_S$. For example, $\mathcal{N}_S(\cap)$ denotes the extension with intersection, and $\mathcal{N}_S(\cap, \pi)$ denotes the extension with intersection and both projections. We will omit the subscript S in $\mathcal{N}_S(F)$ when the precise database schema is not of importance. Two expressions e_1 and e_2 are called equivalent, denoted by $e_1 \equiv e_2$, if they express the same path query, i.e., $e_1(G) = e_2(G)$ for every graph G.

Various interdependencies exist between the nonbasic features [14]:

$$\begin{aligned}
\mathsf{all} &\equiv \mathsf{di} \cup \mathsf{id} \\
\mathsf{di} &\equiv \mathsf{all} - \mathsf{id} \\
e_1 \cap e_2 &\equiv e_1 - (e_1 - e_2) \\
\pi_1(e) &\equiv (e \circ e^{-1}) \cap \mathsf{id} = (e \circ \mathsf{all}) \cap \mathsf{id} = \overline{\pi}_1(\overline{\pi}_1(e)) = \pi_2(e^{-1}) \\
\pi_2(e) &\equiv (e^{-1} \circ e) \cap \mathsf{id} = (\mathsf{all} \circ e) \cap \mathsf{id} \equiv \overline{\pi}_2(\overline{\pi}_2(e)) = \pi_1(e^{-1}) \\
\overline{\pi}_1(e) &\equiv \mathsf{id} - \pi_1(e) \\
\overline{\pi}_2(e) &\equiv \mathsf{id} - \pi_2(e)
\end{aligned}$$

For example, by the third equation, when we add difference, we get intersection for free. The closure of a fragment F by the above equations is denoted by $\overline{F}$. For example, $\overline{\{-, \mathsf{di}\}} = \{-, \mathsf{di}, \mathsf{all}, \cap, \pi, \overline{\pi}\}$. Clearly F and $\overline{F}$ are equivalent in expressive power. This closure notation will be used extensively in what follows.

[2] Some of our results can be refined to fragments containing just one of the two projections or coprojections, but for others this remains a technical open problem [25].

2.3 Conjunctive Queries

To introduce conjunctive queries (CQs) we switch over to another perspective for instances. Again, let V be some fixed infinite universe of data elements V and let R be a relation name in S of arity n. An R-fact is an expression of the form $R(a_1, \ldots, a_n)$ where $a_i \in V$ for $i = 1, \ldots, n$. An R-instance I is a finite set of R-facts. More generally, an instance I of a database schema S is a union $\bigcup_{R \in S} I(R)$, where $I(R)$ denotes an R-instance. This definition for instances corresponds to the logic-programming perspective [1]. Note that there is a one-to-one correspondence between instances under the logic-programming perspective and the perspective outlined in the beginning of Section 2. Indeed, a tuple t in the relation $I(R)$ can be seen as the R-fact $R(t)$ and vice versa.

We formalize the notion of conjunctive queries as follows. A *conjunctive query* is an expression of the form $Q : H \leftarrow B$ where the *head* H is a tuple of variables and the *body* B is a set of atoms over S. An *atom* is an expression of the form $R(v_1, \ldots, v_n)$ where $R \in S$ and $v_1, \ldots, v_n$ are variables. We denote the set of conjunctive queries over S with CQ_S. When the databases schema S is not of importance we will omit the S subscript and write CQ instead. For a conjunctive query Q, H_Q denotes the head and B_Q denotes the body of Q. We assume that our queries are *safe*, i.e., the variables in the head are present somewhere in the body. Semantically, for every instance I over S, $Q(I)$ is defined as:

$$\{f(H_Q) \mid f \text{ is a homomorphism from } Q \text{ into } I\}.$$

Here, a homomorphism f from Q into I is a function on the variables in H_Q and B_Q to $\mathrm{adom}(I)$ such that $f(B_Q) \subseteq I$. Since our queries are safe, and thus all the variables of H_Q are present in B_Q we also write that f is a homomorphism from B_Q into I. Interchangeably, we write that B_Q maps into I.

Remark 1 It is convenient to assume that variables are data elements in V. Then, we can use the body of a conjunctive query as a database instance. As a consequence, an R-atom can then be thought of as an R-fact.

For every two queries Q_1 and Q_2, we write $Q_1 \sqsubseteq Q_2$ if $Q_1(I) \subseteq Q_2(I)$ for every database instance I over S. When Q_1 and Q_2 are conjunctive queries, it is well know that $Q_1 \sqsubseteq Q_2$ iff $H_{Q_1} \in Q_2(B_{Q_1})$.

3 Boolean query modalities

A *boolean query* over a database schema S is a mapping from instances of S to $\{true, false\}$. As argued in the Introduction, boolean queries can be naturally expressed in terms of the emptiness, or the nonemptiness, of an ordinary query, or by the containment of the results of two queries. Using these three base modalities we can associate an array of boolean query families to any family of queries $\mathcal{F}$ on a common database schema S:

family of boolean queries	expressible in the form	with
$\mathcal{F}^{=\emptyset}$	$q = \emptyset$	$q \in \mathcal{F}$
$\mathcal{F}^{\neq\emptyset}$	$q \neq \emptyset$	$q \in \mathcal{F}$
$\mathcal{F}^{\subseteq}$	$q_1 \subseteq q_2$	$q_1, q_2 \in \mathcal{F}$

For $\mathcal{F}^{\subseteq}$, it is understood that only two queries of the same arity can form a containment boolean query.

When working in the algebra of binary relations, for any fragment F of non-basic features, we abbreviate $\mathcal{N}(F)^{=\emptyset}$, $\mathcal{N}(F)^{\neq\emptyset}$ and $\mathcal{N}(F)^{\subseteq}$ by $F^{=\emptyset}$, $F^{\neq\emptyset}$ and $F^{\subseteq}$, respectively.

Obviously, these are by no means the only way to express boolean queries. We could, for example, allow boolean connectives within a family of boolean queries. Indeed, we can consider boolean queries of the form $q_1 \neq \emptyset \wedge \ldots \wedge q_n \neq \emptyset$ where $q_i \neq \emptyset \in \mathcal{F}^{\neq\emptyset}$ where $i = 1, \ldots, n$. Furthermore, we could even combine two different families of boolean queries by using boolean connectives. For example, we can consider boolean queries of the form $q_1 \neq \emptyset \wedge q_2 \subseteq q_3$ where $q_1 \neq \emptyset \in \mathcal{F}^{\neq\emptyset}$ and $q_2 \subseteq q_3 \in F^{\subseteq}$. Our goal in this paper is to propose a framework along which we can investigate and compare different ways of expressing boolean queries.

4 Comparing the modalities

The goal of this section is to compare $\mathcal{F}^{=\emptyset}$, $\mathcal{F}^{\neq\emptyset}$ and $\mathcal{F}^{\subseteq}$, for a fixed family of queries (modeling a query language) $\mathcal{F}$. This amounts to making six comparisons, but we can immediately get one of them out of the way by noting that $\mathcal{F}^{=\emptyset}$ is the negation of $\mathcal{F}^{\neq\emptyset}$. Formally, for a boolean query q, we define its negation $\neg q$ naturally as $(\neg q)(I) = \neg q(I)$, where $\neg \mathit{true} = \mathit{false}$ and $\neg \mathit{false} = \mathit{true}$. For a family of boolean queries $\mathcal{A}$, we define its negation $\neg\mathcal{A}$ as $\{\neg q \mid q \in \mathcal{A}\}$.

Now clearly $\mathcal{A} \subseteq \mathcal{B}$ if and only if $\neg\mathcal{A} \subseteq \neg\mathcal{B}$. Hence, we only have to investigate whether $\mathcal{F}^{=\emptyset} \subseteq \mathcal{F}^{\neq\emptyset}$; the other direction $\mathcal{F}^{\neq\emptyset} \subseteq \mathcal{F}^{=\emptyset}$ then directly follows. This amounts to investigating when the emptiness modality is *closed under negation*. Formally, a family $\mathcal{B}$ of boolean queries is called closed under negation if $\neg\mathcal{B} \subseteq \mathcal{B}$ (or, equivalently, $\mathcal{B} \subseteq \neg\mathcal{B}$). Note that we define closure under negation semantically, it thus applies to any family of boolean queries, so it is not a syntactic definition as it would apply to a query language. (e.g., formulas that do not use certain operators or connectives like difference or logical negation)

We first identify query features that enable the expression of one base modality in terms of another one. We also identify general properties that reflect the absence of these query features, notably, the properties of monotonicity and additivity. We then observe how these properties indeed prevent going from one modality to another.

The announced query features are summarized in the following theorem. We leave out the comparison $\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{\neq\emptyset}$, since we know of no other general way of going from containment to nonemptiness than via emptiness $\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{=\emptyset} \subseteq \mathcal{F}^{\neq\emptyset}$. This leaves four comparisons, dealt with in the following theorem. We refer to the notions introduced in Section 2.1.

Theorem 1 *Let $\mathcal{F}$ be a family of queries.*

1. *$\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{=\emptyset}$ if $\mathcal{F}$ is closed under set difference ($-$).*
2. *$\mathcal{F}^{=\emptyset} \subseteq \mathcal{F}^{\neq\emptyset}$ if there exists k such that $\mathcal{F}$ is closed under*
   - *k-ary complementation, and*
   - *k-ary cylindrification.*
3. *$\mathcal{F}^{\neq\emptyset} \subseteq \mathcal{F}^{\subseteq}$ if*

- *$\mathcal{F}$ contains a never-empty query (one that returns nonempty on every instance), and*
- *$\mathcal{F}$ is closed under tests, or $\mathcal{F}$ is closed under k-ary cylindrification for some k.*

4. *$\mathcal{F}^{=\emptyset} \subseteq \mathcal{F}^{\subseteq}$ if $\mathcal{F}$ contains the* empty *query which always outputs the empty relation.*

Proof 1. $q_1 \subseteq q_2$ is expressed by $q_1 - q_2 = \emptyset$.
2. $q = \emptyset$ is expressed by $\gamma_k(q)^c \neq \emptyset$.
3. Let p be a never-empty query. Then $q \neq \emptyset$ is expressed by $p \subseteq (p \,\text{if}\, q)$ as well as by $\gamma_k(p) \subseteq \gamma_k(q)$.
4. $q = \emptyset$ is expressed by $q \subseteq$ empty. □

Obviously, the above theorem only provides sufficient conditions under which we can go from one modality to another. Since the conditions hold for any general family $\mathcal{F}$, we cannot expect the literal converses of these statements to hold in general. Indeed, one could always concoct an artificial family $\mathcal{F}$ that is not closed under set difference but for which $\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{=\emptyset}$.

Example 3 Over a schema with two unary relation names R and S, let $\mathcal{F}$ be the set of queries

$$\text{if } C \text{ then } e_1 \text{ else } e_2$$

with C finite boolean combinations of expressions $h_i \subseteq h_j$ and e_1, e_2, h_i, h_j are taken from $\{\emptyset, R, S, R \cup S\}$. It can be verified that $\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{=\emptyset}$, and that $\mathcal{F}$ is not closed under difference. □

Our approach to still find a kind of converse of Theorem 1, is to come up with general semantic properties of the queries in a family that would prevent the sufficient conditions to hold. We can then proceed to show that the different modalities become incomparable under these properties.

More concretely, we can observe two main themes in the sufficient conditions: *negation*, in the forms of set difference and complementation, and *global access to the database*, in the forms of cylindrification and tests. A well-known semantic property of queries that runs counter to negation is *monotonicity*. Global access is an intuitive notion. As a formal property that intuitively prevents global access, we propose *additivity*.

4.1 Monotonicity

A query q is *monotone* if $I \subseteq J$ implies $q(I) \subseteq q(J)$, where $I \subseteq J$ means that $I(R) \subseteq J(R)$ for each relation name R. In Theorem 1, we have seen that closure under complementation or set difference, which typically destroys monotonicity, is instrumental for the emptiness modality to be closed under negation, as well as the containment modality to be subsumed by emptiness. We next show that both fail under monotonicity.

The first failure is the strongest:

Lemma 1 *Let* MON *denote the family of monotone queries. The only boolean queries in* $\text{MON}^{=\emptyset} \cap \text{MON}^{\neq\emptyset}$ *are the constant* true *and* false *queries.*

Proof Suppose for the sake of contradiction that a nonconstant boolean query $q = \emptyset \in \text{MON}^{=\emptyset}$ is also in $\text{MON}^{\neq\emptyset}$. Then, there exists $q' \in \text{MON}^{\neq\emptyset}$ such that for any instance I, $q(I) = \emptyset$ iff $q'(I) \neq \emptyset$. Since q is nonconstant, there exist two instances I and J over S such that $q(I) \neq \emptyset$ and $q(J) = \emptyset$. Then, $q'(I) = \emptyset$ and $q'(J) \neq \emptyset$. Thus since q and q' are both in MON, we have $\emptyset \neq q(I) \subseteq q(I \cup J)$ and $\emptyset \neq q'(J) \subseteq q'(I \cup J)$. Therefore, $q(I \cup J) \neq \emptyset$ and $q'(I \cup J) \neq \emptyset$ which is clearly a contradiction. □

For example, let q_1 be the query that always returns the unary empty set. Let q_2 be the unary query that always returns the active domain. Both q_1 and q_2 are monotone. The constant true query is expressed by $q_1 = \emptyset$, as well as by $q_2 \neq \emptyset$.

As a direct corollary of Lemma 1, we obtain:

Proposition 1 *Let $\mathcal{F}$ be a family of monotone queries. As soon as $\mathcal{F}^{=\emptyset}$ contains a non-constant query, $\mathcal{F}^{=\emptyset} \not\subseteq \mathcal{F}^{\neq\emptyset}$.*

This also implies that for any monotone family of queries $\mathcal{F}$ that contains the empty query, we have $\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{\neq\emptyset}$, since $\mathcal{A}^{=\emptyset} \subseteq \mathcal{A}^{\subseteq}$ for any family of queries $\mathcal{A}$ that contains the empty query. We will apply Proposition 1 to conjunctive queries in Section 4.3

We next turn to the failure of going from containment to emptiness. Whenever q is monotone, the boolean query $q = \emptyset$ is antimonotone (meaning that if $q(I) =$ false and $I \subseteq J$, also $q(J) =$ false). However, a boolean containment query is typically not antimonotone. The following straightforward result gives two examples.

Proposition 2 *Let $\mathcal{F}$ be a family of monotone queries over a database schema S.*

1. *If S contains two distinct relation names R and T of the same arity, and the two queries R and T belong to $\mathcal{F}$, then $\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{=\emptyset}$. This is shown by the boolean query $R \subseteq T$.*
2. *If R is a binary relation name in S and the two queries $R \circ R$ and R belong to $\mathcal{F}$, then $\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{=\emptyset}$.*

Proof 1. The query $R \subseteq T$ is not antimonotone.
2. The query "R is transitive", or $R \circ R \subseteq R$, is not antimonotone. □

4.2 Additivity

A query q is *additive* if for any two instances I and J such that $\text{adom}(I)$ and $\text{adom}(J)$ are disjoint, $q(I \cup J) = q(I) \cup q(J)$. Additive queries (also known as "queries distributing over components") have been recently singled out as a family of queries that are well amenable to distributed computation [2]. Indeed, additivity means that a query can be separately computed on each connected component, after which all the subresults can simply be combined by union to obtain the final result.

Additivity and monotonicity are orthogonal properties. For example, the additive queries are closed under set difference. Thus, additive queries may involve negation and need not be monotone. On the other hand, computing the Cartesian product of two relations is monotone but not additive.

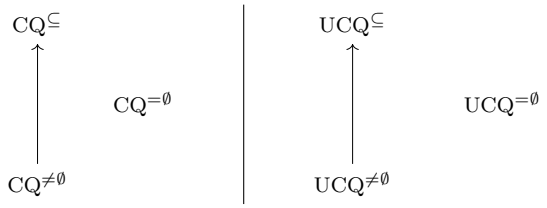

Fig. 1 These diagrams visualize Theorem 2. The arrows in the diagrams depict the subsumption relation of boolean query families.

Both cylindrification and tests run counter to additivity. For example, just computing $\mathrm{adom}(I) \times \mathrm{adom}(I)$ is not additive. Also tests of the form (q_1 if q_2) are not additive, since testing if q_2 is nonempty takes part in the entire instance, across connected components. We have seen that cylindrification (together with complementation) can be used to close the emptiness modality under negation; moreover, cylindrification or tests suffice to move from nonemptiness to containment. We next show that this all fails under additivity.

The following lemma is of a similar nature as Lemma 1.

Lemma 2 *Let* ADD *denote the family of additive queries. The only boolean queries in* $\mathrm{ADD}^{\neq\emptyset} \cap \mathrm{ADD}^{\subseteq}$ *are the constant* true *and* false *queries.*

Proof Suppose for the sake of contradiction that a nonconstant boolean query $q \neq \emptyset \in \mathrm{ADD}^{\neq\emptyset}$ is also in $\mathrm{ADD}^{\subseteq}$. Then, there exist two k-ary queries q_1 and q_2 in ADD such that for any instance I we have $q_1(I) \subseteq q_2(I)$ iff $q(I) \neq \emptyset$. Since q is nonconstant, there exist two instances I and J such that $q(I) \neq \emptyset$ and $q(J) = \emptyset$. Hence $q_1(I) \subseteq q_2(I)$ and $q_1(J) \not\subseteq q_2(J)$. We may assume that $\mathrm{adom}(I)$ and $\mathrm{adom}(J)$ are disjoint since queries are defined to be generic. Therefore, since q is additive, we have $q(I \cup J) = q(I) \cup q(J) \neq \emptyset$, whence we have $q_1(I \cup J) \subseteq q_2(I \cup J)$. Thus we have $q_1(I) \cup q_1(J) \subseteq q_2(I) \cup q_2(J)$. However, this implies that $q_1(J) \subseteq q_2(J)$ since $q_1(J) \subseteq \mathrm{adom}(J)^k$ and $q_2(I) \subseteq \mathrm{adom}(I)^k$, which is a contradiction. □

As a direct corollary, we obtain:

Proposition 3 *Let $\mathcal{F}$ be a family of additive queries.*

1. *As soon as $\mathcal{F}^{\subseteq}$ contains a non-constant query, $\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{\neq\emptyset}$.*
2. *As soon as $\mathcal{F}^{\neq\emptyset}$ contains a non-constant query, $\mathcal{F}^{\neq\emptyset} \not\subseteq \mathcal{F}^{\subseteq}$ and $\mathcal{F}^{=\emptyset} \not\subseteq \mathcal{F}^{\neq\emptyset}$.*

4.3 Conjunctive queries

In this brief section we compare the three base modalities for the popular languages CQ (conjunctive queries) and UCQ (unions of conjunctive queries). The result is that nonemptiness is strictly subsumed by containment, and that all other pairs of modalities are incomparable (cf. Fig. 1).

Theorem 2 *Let $\mathcal{F}$ be* CQ *or* UCQ*. Then*

1. *$\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{=\emptyset}$ and $\mathcal{F}^{=\emptyset} \not\subseteq \mathcal{F}^{\subseteq}$.*

2. $\mathcal{F}^{=\emptyset} \not\subseteq \mathcal{F}^{\neq\emptyset}$.
3. $\mathcal{F}^{\neq\emptyset} \subseteq \mathcal{F}^{\subseteq}$.
4. $\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{\neq\emptyset}$.

Proof 1. Consider the instance Z where $Z(R) = \{(1, \ldots, 1)\}$ for each relation R. Every query in $\mathcal{F}^{\subseteq}$ returns true on Z, whereas every query in $\mathcal{F}^{=\emptyset}$ returns false.
2. By Proposition 1.
3. By Theorem 1(3). Indeed, a CQ with an empty body is never empty. CQs and UCQs are also closed under tests. Indeed, let q_1 and q_2 be UCQs. Then (q_1 if q_2) is expressed by the UCQ consisting of the following rules. Take a rule r of q_1 and a rule s of q_2. Produce the rule obtained from r by conjoining to the body a variable-renamed copy of the body of s. If q_1 has n rules and q_2 has m rules, we obtain nm rules. In particular, if q_1 and q_2 are CQs, we obtain a single rule so again a CQ.
4. Let R be a relation name in the database schema, and consider the two queries

$$q_1(x, y) \leftarrow R(x, \_, \ldots, \_), R(y, \_, \ldots, \_)$$
$$q_2(x, x) \leftarrow R(x, \_, \ldots, \_).$$

Here, the underscores stand for fresh nondistinguished variables (Prolog notation). Then $q_1 \subseteq q_2$ returns true on an instance I iff the first column of $R(I)$ holds at most one distinct element. This boolean query is not monotone and thus not in $\mathcal{F}^{\neq\emptyset}$. □

Remark 2 In the proof of Theorem 2(4) we make convenient use of repeated variables in the head. For the version of CQs where this is disallowed, the result can still be proven by using

$$q_1(x_1, \ldots, x_k) \leftarrow R(x_1, \ldots, x_k)$$
$$q_2(x_1, \ldots, x_k) \leftarrow R(x_1, \ldots, x_k), R(x_k, \_, \ldots, \_).$$

This does not work if R is unary; if there are two different relation names R and T, we can use

$$q_1(x) \leftarrow R(x, \_, \ldots, \_)$$
$$q_2(x) \leftarrow T(x, \_ \ldots, \_).$$

These arguments only fail when the database schema consists of just one single unary relation name, and we cannot use repeated variables in the head. In this extreme case, both $\mathrm{CQ}^{\subseteq}$ and $\mathrm{CQ}^{\neq\emptyset}$ consist only of the constant true query, so the subsumption becomes trivial. □

4.4 Navigational graph query languages

In this section we compare the three base modalities for the navigational graph query languages introduced in the Preliminaries.

The results are summarized in the following theorem. This theorem can be seen as a version of our earlier Theorem 1, specialized to navigational graph query

language fragments. However, now, every statement is a *characterization*, showing that the sufficient condition is also necessary for subsumption to hold. Particularly satisfying is that, with a few exceptions, almost the entire theorem can be proven from the general results given earlier.

Theorem 3 *Let F be a navigational graph query language fragment.*

1. $F^{\subseteq} \subseteq F^{=\emptyset}$ *if and only if* $- \in F$.
2. $F^{=\emptyset} \subseteq F^{\neq\emptyset}$ *if and only if* $\mathsf{all} \in \overline{F}$ *and* ($- \in F$ *or* $\overline{\pi} \in \overline{F}$).
3. $F^{\neq\emptyset} \subseteq F^{\subseteq}$ *if and only if* $\mathsf{all} \in \overline{F}$.
4. $F^{\subseteq} \subseteq F^{\neq\emptyset}$ *if and only if* $\mathsf{all} \in \overline{F}$ *and* $- \in F$.

Notice that Theorem 3 no longer contains an adapted version for Theorem 1(4). This is because the empty query is in $\mathcal{N}(F)$ for any fragment F by definition, whence $F^{=\emptyset} \subseteq F^{\subseteq}$ always holds. Instead, we now do provide in item 4 an explicit characterization for when the subsumption from containment to nonemptiness holds.

In every part of the above theorem, the if-direction can be seen by showing that $\mathcal{N}(F)$ fulfills the conditions of Theorem 1.

1. This follows immediately from Theorem 1(1).
2. When set difference is present, the binary complementation of q is expressible by $\mathsf{all} - q$. Also the binary cylindrification of q is expressible by $\mathsf{all} \circ q \circ \mathsf{all}$. Hence, Theorem 1(2) readily applies with $k = 2$.
   When set difference is not present, we have coprojection. We can now apply Theorem 1(2) with $k = 1$. We simulate unary relations by subsets of the identity relation id. In particular, the unary cylindrification of q is expressed by $\pi_1(\mathsf{all} \circ q)$ and $\pi_2(q \circ \mathsf{all})$, and unary complement is provided by coprojection.
3. We have already seen how binary cylindrification is expressible using all. Furthermore, all also provides a never-empty query. Hence, Theorem 1(3) readily applies.
4. We have $F^{\subseteq} \subseteq F^{=\emptyset} \subseteq F^{\neq\emptyset}$.

4.4.1 Inexpressibility results

To prove the only-if directions of Theorem 3, we will exhibit inexpressibility results.

For the first part, it is sufficient to show that $F^{\subseteq}$ is not subsumed by $F^{=\emptyset}$ for every fragment F without set difference. Thereto we introduce NoDiff, the largest fragment without set difference, which is defined as $\{\mathsf{di}, {}^{-1}, \cap, \overline{\pi}, {}^{+}\}$. The following lemma establishes the first part of the theorem by exhibiting, for every fragment F, a boolean query in $F^{\subseteq}$ but not in $\text{NoDiff}^{=\emptyset}$.

Lemma 3 *Let R be a relation schema in S. Then the boolean query "R is transitive", formally, $R \circ R \subseteq R$, is neither in* $\text{NoDiff}^{=\emptyset}$ *nor in* $\text{NoDiff}^{\neq\emptyset}$.

Proof Over the single relation name R, consider the complete directed graph on three nodes K_3, and a graph B in the form of a bow tie, i.e., two K_3 copies with one shared node. (Both K_3 and B are displayed in Fig. 2.) There is a self-loop at every node. It is known that K_3 and B are indistinguishable by boolean queries in $\text{NoDiff}^{\neq\emptyset}$ [16, Proposition 5.6(1)]. This implies that both graphs are also indistinguishable by boolean queries in $\text{NoDiff}^{=\emptyset}$. However, K_3 is transitive while B is not. □

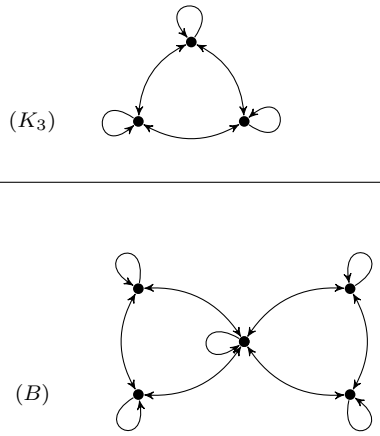

Fig. 2 The complete graph with 3 nodes (K_3) and bow-tie (B) graphs.

The only-if directions of the remaining parts of the theorem all revolve around the fragment NoAll = $\{^{-1}, -, ^{+}\}$, the largest fragment without the full relation all. This fragment lacks the only two features (di and all) that allow to jump from one connected component to another. The following lemma also follows from the additivity of connected stratified Datalog [2]. We, however, also give a direct proof:

Additivity Lemma *Every binary-relation query in* $\mathcal{N}$(NoAll) *is additive.*

Proof Let e be an expression in $\mathcal{N}$(NoAll), and let G and H be graphs such that $\mathrm{adom}(G) \cap \mathrm{adom}(H) = \emptyset$. We must show that $e(G \cup H) = e(G) \cup e(H)$. We proceed by structural induction on e. The case where e is a relation name is trivial and the case where e is id is clear.

If $e = e_1 \circ e_2$, then

$$
\begin{aligned}
&(x,y) \in e_1 \circ e_2(G \cup H)\\
&\text{ iff } \exists z : (x,z) \in e_1(G \cup H) \wedge (z,y) \in e_2(G \cup H)\\
&\overset{*}{\text{iff}} \exists z : (x,z) \in e_1(G) \cup e_1(H)\\
&\qquad\qquad \wedge (z,y) \in e_2(G) \cup e_2(H)\\
&\text{iff } \exists z : ((x,z) \in e_1(G) \vee (x,z) \in e_1(H))\\
&\qquad\qquad \wedge ((z,y) \in e_2(G) \vee (z,y) \in e_2(H))\\
&\overset{**}{\text{iff}} \exists z : ((x,z) \in e_1(G) \wedge (z,y) \in e_2(G))\\
&\qquad\qquad \vee ((x,z) \in e_1(H) \wedge (z,y) \in e_2(H))\\
&\text{iff } (x,y) \in e_1 \circ e_2(G) \vee (x,y) \in e_1 \circ e_2(H).
\end{aligned}
$$

The equivalence marked with a single * follows from the induction hypothesis. Furthermore, the equivalence marked with ** holds because $e_1(G) \subseteq \mathrm{adom}(G)^2$,

$e_1(H) \subseteq \mathrm{adom}(H)^2$, and $\mathrm{adom}(G) \cap \mathrm{adom}(H) = \emptyset$. Indeed, because of this observation we can drop the cases $(x,z) \in e_1(G) \wedge (z,y) \in e_2(H)$ and $(x,z) \in e_1(H) \wedge (z,y) \in e_2(G)$.

If e is of the form e_1^{-1}, then

$$\begin{aligned}
&e_1^{-1}(G \cup H)\\
&= \{(y,x) \in \mathrm{adom}(G \cup H)^2 \mid (x,y) \in e_1(G \cup H)\}\\
&= \{(y,x) \in \mathrm{adom}(G \cup H)^2 \mid (x,y) \in e_1(G) \cup e_1(H)\}\\
&= \{(y,x) \in \mathrm{adom}(G \cup H)^2 \mid (x,y) \in e_1(G)\}\\
&\quad \cup \{(y,x) \in \mathrm{adom}(G \cup H)^2 \mid (x,y) \in e_1(H))\}\\
&= \{(y,x) \in \mathrm{adom}(G)^2 \mid (x,y) \in e_1(G)\}\\
&\quad \cup \{(y,x) \in \mathrm{adom}(H)^2 \mid (x,y) \in e_1(H))\}\\
&= e_1^{-1}(G) \cup e_1^{-1}(H).
\end{aligned}$$

If $e = e_1 \cup e_2$, then

$$\begin{aligned}
e_1 \cup e_2(G \cup H) &= e_1(G \cup H) \cup e_2(G \cup H)\\
&= e_1(G) \cup e_1(H) \cup e_2(G) \cup e_2(H)\\
&= (e_1 \cup e_2)(G) \cup (e_1 \cup e_2)(H).
\end{aligned}$$

If $e = e_1 - e_2$, then

$$\begin{aligned}
e_1 - e_2(G \cup H) &= e_1(G \cup H) - e_2(G \cup H)\\
&= (e_1(G) \cup e_1(H)) - (e_2(G) \cup e_2(H))\\
&= (e_1(G) - e_2(G)) \cup (e_1(H) - e_2(H))\\
&= (e_1 - e_2)(G) \cup (e_1 - e_2)(H).
\end{aligned}$$

If $e = e_1^+$ then $e_1^+(G \cup H) = (e_1(G) \cup e_1(H))^+$ by induction. Now since $\mathrm{adom}(G) \cap \mathrm{adom}(H) = \emptyset$ we also have that $(e_1(G) \cup e_1(H))^+ = e_1^+(G) \cup e_1^+(H)$. □

The Additivity Lemma allows an easy proof for the second and third parts of the theorem, as we next demonstrate. Also the proofs of several later results hinge upon additivity.

For the second part, we must prove that $F^{=\emptyset}$ is not subsumed by $F^{\neq\emptyset}$ for any fragment F without all, as well as any fragment having neither difference nor coprojection. The latter case is clear. Indeed, difference and coprojection are the only two nonmonotone operators. Thus $\mathcal{N}(F)$ is monotone, whence Proposition 1 proves the result.

For a fragment F without all but possibly with difference or coprojection, we have that $\mathcal{N}(F)$ is additive. Hence, Proposition 3 establishes the second as well as the third parts when $\mathsf{all} \notin F$.

Finally, for the fourth part, we must prove that $F^{\subseteq}$ is not subsumed by $F^{\neq\emptyset}$ for any fragment F without all or without set difference. The case without set difference already follows from Lemma 3. The case without all already follows from the second part.

Regular path queries. The fragment $\{^+\}$ corresponds to a well known family of graph queries called regular path queries (RPQ) [12]. Theorem 3 directly tells us that $\mathrm{RPQ}^{=\emptyset} \not\subseteq \mathrm{RPQ}^{\neq\emptyset}$, $\mathrm{RPQ}^{\neq\emptyset} \not\subseteq \mathrm{RPQ}^{\subseteq}$, $\mathrm{RPQ}^{\subseteq} \not\subseteq \mathrm{RPQ}^{=\emptyset}$ and $\mathrm{RPQ}^{\subseteq} \not\subseteq \mathrm{RPQ}^{\neq\emptyset}$.

5 Cross-language comparisons

In the previous section, we have compared different modalities within a given family of queries (query language). Dually, one may investigate how different query languages compare for a given modality. In the context of navigational graph query languages, we have already done this research [29,27].

The next step, then, is to see how different query languages relate when using different modalities. This question is particularly interesting since nonemptiness is the standard modality for expressing boolean queries and containment is a fundamentally different but also very natural modality. Then it is interesting, e.g., to try to understand to what extent the containment modality, using some language $\mathcal{F}_2$, can be used to express nonemptiness queries using some other language $\mathcal{F}_1$. For example, $R \circ R \neq \emptyset$ in $\{\}^{\neq\emptyset}$ is expressed by $\mathsf{all} \subseteq \mathsf{all} \circ R \circ R \circ \mathsf{all}$ in $\{\mathsf{all}\}^{\subseteq}$.

In this paper, we only do this in the context of navigational graph query languages. First, we compare containment to (non)emptiness; and nonemptiness to emptiness.

Theorem 4 *Let F_1 and F_2 be fragments. Then, we have:*

1. $F_1^{\subseteq} \subseteq F_2^{=\emptyset}$ *iff* $F_1^{\subseteq} \subseteq F_2^{\subseteq}$ *and* $F_2^{\subseteq} \subseteq F_2^{=\emptyset}$;
2. $F_1^{\subseteq} \subseteq F_2^{\neq\emptyset}$ *iff* $F_1^{\subseteq} \subseteq F_2^{\subseteq}$ *and* $F_2^{\subseteq} \subseteq F_2^{\neq\emptyset}$;
3. $F_1^{\neq\emptyset} \subseteq F_2^{=\emptyset}$ *iff* $F_1^{\neq\emptyset} \subseteq F_2^{\neq\emptyset}$ *and* $F_2^{\neq\emptyset} = F_2^{=\emptyset}$.

Proof The if directions hold due to the transitivity of $\subseteq$. For the only if direction, we consider the contrapositives. We then have:

1. If $F_2^{\subseteq} \not\subseteq F_2^{=\emptyset}$, then $- \notin F_2$ by Theorem 3. Hence, $R^2 \subseteq R$ is not in $F_2^{=\emptyset}$ by Lemma 3.
   Conversely, If $F_2^{\subseteq} \subseteq F_2^{=\emptyset}$, then $- \in F_2$ by Theorem 3, whence $F_2^{=\emptyset} = F_2^{\subseteq}$;
2. If $F_2^{\subseteq} \not\subseteq F_2^{\neq\emptyset}$, then $- \notin F_2$ or $\mathsf{all} \notin F_2$ by Theorem 3. If $- \notin F_2$, then $R^2 \subseteq R$ is not in $F_2^{\neq\emptyset}$ by Lemma 3. On the other hand, if $\mathsf{all} \notin \overline{F_2}$, then $\mathcal{N}(F_2)$ is additive by the additivity lemma. Hence, $R_2 \subseteq R$ is not in $F_2^{\neq\emptyset}$.
   Conversely, if $F_2^{\subseteq} \subseteq F_2^{\neq\emptyset}$, then $-, \mathsf{all} \in \overline{F_2}$ by Theorem 3, whence $F_2^{\neq\emptyset} = F_2^{\subseteq}$.
3. If $F_2^{\neq\emptyset} \neq F_2^{=\emptyset}$, then $-, \overline{\pi} \notin \overline{F_2}$ or $\mathsf{all} \notin \overline{F_2}$ by 3. In the former case, $\mathcal{N}(F_2)$ is monotone, whence the result directly follows from Proposition 1. In the latter case, $\mathcal{N}(F_2)$ is additive, whence the result directly follows from Lemma 2. □

Next, we turn our attention to when the inclusion $F_1^{\neq\emptyset} \subseteq F_2^{\subseteq}$ holds, for different navigational graph query language fragments F_1 and F_2.

Example 4 For a positive example, consider the query $R^2 \circ R^{-1} \circ R^2 \neq \emptyset$ in $\{^{-1}\}^{\neq\emptyset}$. This query is expressed by $\mathsf{all} \subseteq \mathsf{all} \circ \pi_1(R^2 \circ \pi_2(\pi_1(R^2) \circ R)) \circ \mathsf{all}$ in $\{\pi, \mathsf{all}\}^{\subseteq}$. For a negative example, we can show that $R^2 \circ R^{-1} \circ R^2 \neq \emptyset$ is not in $\{\mathsf{all}\}^{\subseteq}$. □

5.1 Comparing nonemptiness to containment

Similar to the structure of Theorem 4, whenever we can move from F_1 to F_2 staying with the nonemptiness modality, i.e., $F_1^{\neq\emptyset} \subseteq F_2^{\neq\emptyset}$, and moreover, we can switch from nonemptiness to containment within F_2, i.e., $F_2^{\neq\emptyset} \subseteq F_2^{\subseteq}$, we obviously obtain $F_1^{\neq\emptyset} \subseteq F_2^{\subseteq}$ by transitivity. Actually, our conjecture is that nothing else can happen:

Conjecture 1 Let F_1 and F_2 be fragments. If $F_1^{\neq\emptyset} \subseteq F_2^{\subseteq}$, then $F_2^{\neq\emptyset} \subseteq F_2^{\subseteq}$ and $F_1^{\neq\emptyset} \subseteq F_2^{\neq\emptyset}$.

We can prove large parts of this conjecture; the only open case revolves around the fragments $F_1 = \{\pi\}$ and $F_2 \subseteq \{\mathsf{di}, \mathsf{all}, {}^{-1}, {}^{+}\}$. In particular, if one could show that

$$\{\pi\}^{\neq\emptyset} \not\subseteq \{\mathsf{di}, {}^{-1}, {}^{+}\}^{\subseteq}$$

then Conjecture 1 would be entirely resolved.

It is sufficient to prove the conjecture under the following two assumptions:

- If $F_2^{\neq\emptyset} \not\subseteq F_2^{\subseteq}$, our proof of Theorem 3(3) actually implies $\mathcal{N}^{\neq\emptyset} \not\subseteq F_2^{\subseteq}$ (recall that $\mathcal{N}$ is the most basic fragment). Hence, certainly $F_1^{\neq\emptyset} \not\subseteq F_2^{\subseteq}$, so the conjecture is void in this case. Thus, we may assume that $F_2^{\neq\emptyset} \subseteq F_2^{\subseteq}$, i.e., that all is present in F_2.
- If moreover $-$ is in F_2, then $F_2^{\subseteq} = F_2^{\neq\emptyset}$, and the conjecture becomes trivial again. Thus, we may assume that $-$ is not in F_2.

Under the above assumptions we propose to prove the conjecture by its contrapositive. So we assume $F_1^{\neq\emptyset} \not\subseteq F_2^{\neq\emptyset}$ and try to establish $F_1^{\neq\emptyset} \not\subseteq F_2^{\subseteq}$. Now the given $F_1^{\neq\emptyset} \not\subseteq F_2^{\neq\emptyset}$ has been precisely characterized in our previous work [29]. We refer to the paper [29], which shows that $F_1^{\neq\emptyset} \not\subseteq F_2^{\neq\emptyset}$ can only happen in the following cases:

Transitive closure: Transitive closure is present in $\overline{F}_1$ but not in $\overline{F}_2$, and either the database schema has at least two relation names, or $\overline{F}_1$ contains at least one of $\cap$, $\overline{\pi}$ or ${}^{-1}$.

Converse: Converse is in $\overline{F}_1$ but not in $\overline{F}_2$, and
(a) $\cap$ is in $\overline{F}_1$;
(b) ${}^{+}$ is in $\overline{F}_1$; or
(c) $F_1 \subseteq \{{}^{-1}, \mathsf{di}, \mathsf{all}, \pi, \overline{\pi}\}$ and $F_2 \subseteq \{\mathsf{all}, \mathsf{di}, {}^{+}\}$.

Intersection, difference, diversity, coprojection, or projection: One of these is in $\overline{F}_1$ but not in $\overline{F}_2$.

We can deal completely with all cases, except for projection, which we will discuss last. We devote one section to each feature.

5.1.1 Intersection

The largest fragment for F_2 we need to consider is NoInt $= \{\mathsf{di}, \overline{\pi}, {}^{-1}, {}^{+}\}$ ("no intersection"). We show that the query $R \cap \mathsf{id} \neq \emptyset$ ("the graph has self-loops") is not in NoInt$^{\subseteq}$. To prove this proposition it suffices to reason on the two finite graphs in Fig. 3.

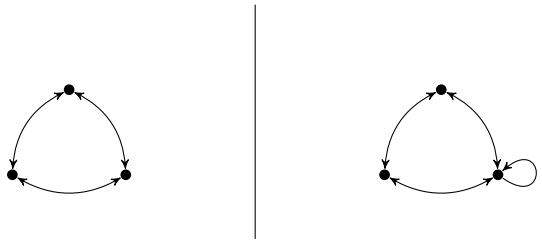

Fig. 3 The graphs used to prove Proposition 4.

Lemma 4 *Let e be an expression in $\mathcal{N}(\mathsf{di}, {}^{-1})$. On the graphs A_1 (left) and A_2 (right) shown in Fig. 3, e is equivalent to $\emptyset$, id, di, R or all simultaneously.*

Proof We prove this lemma by structural induction on e. For id, di and R this is trivial. For R^{-1} note that A_1 and A_2 are symmetrical.

Suppose $e = e_1 \cup e_2$. Then the only troublesome cases are:

- $e_1 = \mathsf{id}$ and $e_2 = R$ or vice versa. Here, $e_1 \cup e_2(A_1) = \mathsf{all}(A_1)$ and $e_1 \cup e_2(A_2) = \mathsf{all}(A_2)$.
- $e_1 = R$ and $e_2 = \mathsf{di}$ or vice versa. Here, $e_1 \cup e_2(A_1) = R(A_1)$ and $e_1 \cup e_2(A_2) = R(A_2)$.

Suppose $e = e_1 \circ e_2$. Since composing with $\emptyset$ results in $\emptyset$, and composing with id does nothing, we may focus on $e_1, e_2 \in \{\mathsf{di}, R, \mathsf{all}\}$. It is clear that $R \cap \mathsf{di}(A_1) \subseteq e_i(A_1)$ and $R \cap \mathsf{di}(A_2) \subseteq e_i(A_2)$. Hence $(R \cap \mathsf{di}) \circ (R \cap \mathsf{di})(A_1) \subseteq e_1 \circ e_2(A_1)$ and $(R \cap \mathsf{di}) \circ (R \cap \mathsf{di})(A_2) \subseteq e_1 \circ e_2(A_2)$. Therefore, since $(R \cap \mathsf{di}) \circ (R \cap \mathsf{di})(A_1) = \mathsf{all}(A_1)$ and $(R \cap \mathsf{di}) \circ (R \cap \mathsf{di})(A_2) = \mathsf{all}(A_2)$, we obtain $e_1 \circ e_2(A_1) = \mathsf{all}(A_1)$ and $e_1 \circ e_2(A_2) = \mathsf{all}(A_2)$. □

We are now ready to establish the separation.

Proposition 4 *Let R be a relation schema in S. Then the boolean query "R contains a self-loop", formally, $R \cap \mathsf{id} \neq \emptyset$, is not in* $\mathrm{NoInt}^{\subseteq}$.

Proof Let us denote the query $R \cap \mathsf{id} \neq \emptyset$ with Q. Suppose for the sake of contradiction that $e_1 \subseteq e_2$ expresses Q. We will only work with A_1 (left) and A_2 (right) displayed in Fig. 3. We know that $Q(A_2)$ is false, thus $e_1(A_1) \not\subseteq e_2(A_1)$, whence $e_2(A_1) \neq \mathsf{all}(A_1)$. Since we only work on A_1 and A_2, we know that $^+$ can be replaced by unions of compositions. Moreover, since A_1 and A_2 are complete up to the self-loops, we know that the $\overline{\pi}(h)$ for every expression $h \in \mathrm{NoInt}$ is always empty or the identity. Thus, we may assume that e_1 and e_2 are expressions in $\mathcal{N}(\mathsf{di}, {}^{-1})$. By Lemma 4, we know that $e_2(A_1)$ is $\emptyset(A_1)$, $\mathsf{id}(A_1)$, $\mathsf{di}(A_1)$, $R(A_1)$ or $\mathsf{all}(A_1)$.

If $e_2(A_1) = \emptyset$, then $e_2(A_2) = \emptyset$ by Lemma 4. Moreover, since $e_1(A_1) \not\subseteq e_2(A_1)$, it must be that $e_1(A_1)$ is $\mathsf{id}(A_1)$, $R(A_1)$, $\mathsf{di}(A_1)$ or $\mathsf{all}(A_1)$. Hence by Lemma 4, also $e_1(A_2) \neq \emptyset$, which contradictions that $Q(A_2)$ is true.

If $e_2(A_1) = \mathsf{id}(A_1)$, then $e_2(A_2) = \mathsf{id}(A_2)$ by Lemma 4. Since $e_1(A_1) \not\subseteq e_2(A_2)$, we know that $e_1(A_1)$ equals $R(A_1)$, $\mathsf{di}(A_1)$ or $\mathsf{all}(A_1)$. Hence, $e_1(A_2)$ also equals $R(A_1)$, $\mathsf{di}(A_1)$ or $\mathsf{all}(A_1)$ by the same lemma. This contradicts that $Q(A_2)$ is true.

If $e_2(A_1) = R(A_1) = \mathsf{di}(A_1)$, then $e_2(A_2)$ equals $\mathsf{di}(A_2)$ or $R(A_2)$ by Lemma 4. Since $e_1(A_1) \not\subseteq e_2(A_1)$, it must be that $e_1(A_1) = \mathsf{id}(A_1)$ or $\mathsf{all}(A_1)$ by Lemma. 4.

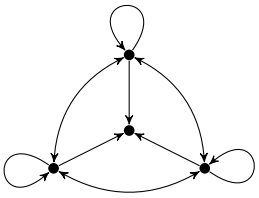

Fig. 4 Graph used to establish separation in Section 5.1.4 for the coprojection when comparing nonemptiness to containment for different languages.

Thus, we also have $e_1(A_2) = \mathsf{id}(A_2)$ or $\mathsf{all}(A_2)$, which contradicts that $Q(A_2)$ is true. □

5.1.2 Difference

The largest fragment we have to consider is NoDiff $= \{\mathsf{di}, {}^{-1}, \cap, \overline{\pi}, {}^{+}\}$. Let B be the bow-tie and K_3 be the complete graph displayed in Fig. 2. The proof of Proposition VI.1 in [27] shows that.

Lemma 5 ([27]) *Any boolean query in* $\text{NoDiff}^{\subseteq}$ *cannot be true on* B *and false on* K_3.

This shows that $R^2 - R \neq \emptyset$ ("the graph is not transitive") is not in $\text{NoDiff}^{\subseteq}$, since it is true on B and false on K_3.

5.1.3 Diversity

The largest fragment we have to consider is $\{\mathsf{all}, {}^{-1}, \overline{\pi}, \cap, {}^{+}\}$. Let id_1 be the graph that consists of a single self-loop. The proof of Proposition IX.1 in [27] shows that id_1 and K_3 are indistinguishable in $\{\mathsf{all}, {}^{-1}, \overline{\pi}, \cap, {}^{+}\}^{\subseteq}$. As a direct corollary, we obtain that $\mathsf{di} \neq \emptyset$ ("the graph has at least two nodes") is not in $\{\mathsf{all}, {}^{-1}, \overline{\pi}, \cap, {}^{+}\}^{\subseteq}$.

5.1.4 Coprojection

The largest fragment we have to consider is $\{\mathsf{di}, {}^{-1}, \cap, {}^{+}\}$. Let G be the graph in Fig. 4. In the proof of Proposition IV.1 in [27] it is shown that every query in $\{\mathsf{di}, {}^{-1}, \cap, {}^{+}\}^{\subseteq}$ cannot be true on G and false on K_3 simultaneously. As a direct corollary we obtain that $\overline{\pi}_1(R) \neq \emptyset$ ("the graph has at least one sink node") is not in $\{\mathsf{di}, {}^{-1}, \cap, {}^{+}\}^{\subseteq}$ since it is true on G and false on K_3.

5.1.5 Transitive Closure

From our earlier work [29] we know that $F_1^{\neq\emptyset}$ can express some query not expressible in first-order logic (FO), whereas $F_2^{\subseteq}$ is clearly subsumed by FO.

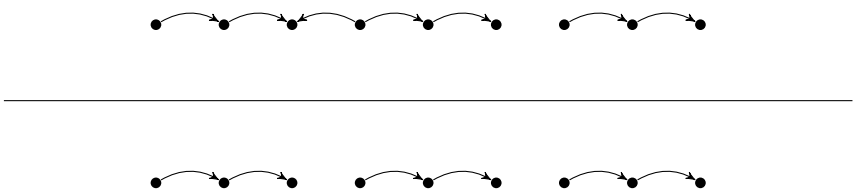

Fig. 5 Graphs used in the proof of Proposition 5.

5.1.6 Converse

The largest fragment without converse is NoConv $= \{\mathsf{di}, \pi, {}^+, -\}$. Since NoConv has both all and $-$, we have $\text{NoConv}^{\neq\emptyset} = \text{NoConv}^{\subseteq}$. Now in case (a), we already know [16, Proposition 6.6] that the query $(R^2 \circ R^{-1} \circ R) \cap R \neq \emptyset$ is not in $\text{NoConv}^{\neq\emptyset}$. In case (b), we already know [29, Proposition 5.4] that $R^2 \circ (R \circ R^{-1})^+ \circ R^2 \neq \emptyset$ is not in $\text{NoConv}^{\neq\emptyset}$. The following lemma settles case (c):

Proposition 5 *Let R be a relation name. The query $R^2 \circ R^{-1} \circ R^2 \neq \emptyset$ is not in $\{\mathsf{di}, {}^+\}^{\subseteq}$.*

Proof Let Q be the query $R^2 \circ R^{-1} \circ R^2 \neq \emptyset$. Let G_1 be the top and G_2 be the bottom graph in Fig. 5. Since our graphs are finite, the set $E = \{(e(G_1), e(G_2)) \mid e \in \mathcal{N}(\mathsf{di}, {}^+)\}$ is finite. Hence, it can be computed by a terminating computer program:

1. Start with the set $E = \{(R(G_1), R(G_2)), (\mathsf{di}(G_1), \mathsf{di}(G_2)), (\mathsf{id}(G_1), \mathsf{id}(G_2))\}$
2. For every two pairs (A, B), (C, D) in E, add $(A \cup C, B \cup D)$ and $(A \circ C, B \circ D)$ to E;
3. For every pair (A, B) in E, add (A^+, B^+) to E;
4. Repeat step 2 and 3 until E no longer changes.

Thus, we can also compute $\{(e_1 \subseteq e_2(G_1), e_1 \subseteq e_2(G_2)) \mid e_1, e_2 \in \mathcal{N}(\mathsf{di}, {}^+)\}$. We have verified that $(\mathit{true}, \mathit{false})$ is not in this set. Therefore, Q is not in $\{\mathsf{di}, {}^+\}^{\subseteq}$ since Q is *true* on G_1 and *false* on G_2. □

5.1.7 Projection

In the case of projection, the largest fragment for F_2 we need to consider is $\{\mathsf{di}, {}^{-1}, {}^+\}$. We would like to show that $\{\pi\}^{\neq\emptyset} \not\subseteq \{\mathsf{di}, {}^{-1}, {}^+\}^{\subseteq}$.

We already know [16] that there are queries in $\{\pi\}^{\neq\emptyset}$ but not in $\{\mathsf{di}, {}^{-1}, {}^+\}^{\neq\emptyset}$. Furthermore, note that queries in $\{\pi\}^{\neq\emptyset}$ are always monotone. Hence, if we could show that monotone queries in $\{\mathsf{di}, {}^{-1}, {}^+\}^{\subseteq}$ are always in $\{\mathsf{di}, {}^{-1}, {}^+\}^{\neq\emptyset}$, the conjecture would be proved.

We can give a partial answer for the union-free fragment of $\{\mathsf{all}, {}^{-1}\}$.

Theorem 5 *There is a boolean query in $\{\pi\}^{\neq\emptyset}$ that is not in $\mathcal{F}_2^{\subseteq}$, where $\mathcal{F}_2$ is the union-free sublanguage of $\mathcal{N}(\mathsf{all}, {}^{-1})$.*

Proof Path queries expressed in the union-free fragment of $\mathcal{N}(\mathsf{all}, {}^{-1})$ are expressible as conjunctive queries, where the conjunctive queries might be unsafe (an *unsafe* conjunctive query has variables in its head that do not occur in its body). Elsewhere [26] we have shown the following result:

> For any database schema S, $\mathrm{CQ}_S^{\subseteq} \cap \mathrm{MON} = \mathrm{CQ}_S^{\neq\emptyset}$. Specifically, every monotone query $Q_1 \subseteq Q_2$, where Q_1 and Q_2 are CQs, is equivalent to a query of the form $((() \leftarrow B) \neq \emptyset)$, where B is empty or B consists of some of the connected components of B_{Q_2}.

This directly implies a similar preservation result for $\mathcal{F}_2$, i.e., $\mathcal{F}_2^{\subseteq} \cap \mathrm{MON} = \mathcal{F}_2^{\neq\emptyset}$ ($\star$). As mentioned above, we already know [16] that there is a query Q in $\{\pi\}^{\neq\emptyset}$ that is not in $\{\mathsf{di}, {}^{-1}, {}^{+}\}^{\neq\emptyset}$, whence Q is not in $\mathcal{F}_2^{\neq\emptyset}$ either. Therefore, Q is not in $\mathcal{F}_2^{\subseteq} \cap \mathrm{MON}$ by ($\star$). Since Q is monotone, we may conclude that Q is not in $\mathcal{F}_2^{\subseteq}$, as desired. □

It is an interesting challenge to try to extend this corollary to the full fragment $\{\mathsf{di}, {}^{-1}, {}^{+}\}$.

6 Closure under boolean connectives

6.1 Closure under negation

In Section 4 we already observed that the question whether $\mathcal{F}^{=\emptyset}$ is subsumed by $\mathcal{F}^{\neq\emptyset}$ is equivalent to whether $\mathcal{F}^{\neq\emptyset}$ is closed under negation. One may now also wonder about the logical negation of $\mathcal{F}^{\subseteq}$. It turns out, however, that $\mathcal{F}^{\subseteq}$ is seldom closed under negation. We denote $\neg\mathcal{F}^{\subseteq}$ (negation is defined in the beginning of Section 4) by $\mathcal{F}^{\not\subseteq}$.

For all family of queries $\mathcal{F}$, we can infer $\mathcal{F}^{\not\subseteq} \subseteq \mathcal{F}^{\subseteq}$ if $\mathcal{F}^{\not\subseteq} \subseteq \mathcal{F}^{\neq\emptyset} \subseteq \mathcal{F}^{\subseteq}$. The first inequality is equivalent to $\mathcal{F}^{\subseteq} \subseteq \mathcal{F}^{=\emptyset}$. Hence, from Proposition 1 we can infer that the containment modality is closed under negation if $\mathcal{F}$ has set difference, contains a never-empty query, and has tests or cylindrification. An alternative route could be taken using $\mathcal{F}^{\not\subseteq} \subseteq \mathcal{F}^{\neq\emptyset} \subseteq \mathcal{F}^{=\emptyset} \subseteq \mathcal{F}^{\subseteq}$, which can be done if $\mathcal{F}$ has set difference, complementation and cylindrification, and contains the empty query. Both routes suggest that closure under negation for the containment modality requires quite a strong query language. We will confirm this in the paragraphs below by showing that it does not hold for CQs or UCQs (as may be expected), and that it holds only for graph query language fragments that include both set difference and all.

In terms of a general negative result, we can only offer the straightforward inference that $\mathcal{F}^{\not\subseteq} \not\subseteq \mathcal{F}^{\subseteq}$ whenever $\mathcal{F}$ is additive and contains the empty query, and $\mathcal{F}^{\neq\emptyset}$ contains a non-constant query. Indeed, using the empty query we have $\mathcal{F}^{\neq\emptyset} \subseteq \mathcal{F}^{\not\subseteq}$, and Proposition 3 yields $\mathcal{F}^{\neq\emptyset} \not\subseteq \mathcal{F}^{\subseteq}$, whence we also have $\mathcal{F}^{\not\subseteq} \not\subseteq \mathcal{F}^{\subseteq}$.

Turning to conjunctive queries, $(\mathrm{U})\mathrm{CQ}^{\not\subseteq} \not\subseteq (\mathrm{U})\mathrm{CQ}^{\subseteq}$ follows immediately from the instance Z used in the proof of Theorem 2. On Z, every boolean query in $\mathrm{UCQ}^{\subseteq}$ returns true, whereas the constant false query is easily expressed in $\mathrm{CQ}^{\not\subseteq}$.

For navigational graph query language fragments F, we have the following characterization.

Theorem 6 *Let F be a fragment. Then, $F^{\not\subseteq} \subseteq F^{\subseteq}$ iff $\mathsf{all} \in \overline{F}$ and $- \in F$.*

Proof The if-direction follows from the general observation in the beginning of this section.

To prove the only-if direction, recall that $\mathcal{N}(\text{NoAll})$ is an additive query language. Hence, $\text{NoAll}^{\not\supseteq} \not\subseteq \text{NoAll}^{\subseteq}$ also follows from the general observations made in the beginning of the section.

So, the only thing left to show is that $\text{NoDiff}^{\not\supseteq} \not\subseteq \text{NoDiff}^{\subseteq}$. Let B be the bow tie and K_3 be the complete graph with three nodes both displayed in Fig. 2. Notice that $\mathsf{all} \not\subseteq R$ is true on B and false on K_3. By Lemma 5, this is not possible in $\text{NoDiff}^{\subseteq}$. □

6.2 Closure under Conjunction

Closure under conjunction is more interesting. Since we often enforce a set (conjunction) of integrity constraints, or specify logical theories consisting of sets of axioms, it is a natural question to ask if such conjunctions can be written as single boolean queries in the same language.

6.2.1 Navigational graph query languages

Under the emptiness modality, closure under conjunction is trivial, since $(q_1 = \emptyset) \wedge (q_2 = \emptyset)$ is equivalent to $q_1 \cup q_2 = \emptyset$.

Under the nonemptiness modality, we have the following.

Theorem 7 *Let F be a fragment. Then $F^{\neq\emptyset}$ is closed under conjunction if and only if either $\mathsf{all} \in \overline{F}$, or the database schema S consists of a single binary relation name and $F \subseteq \{^+\}$.*

Proof If $\overline{F}$ has all, then we can directly express $(e_1 \neq \emptyset) \wedge (e_2 \neq \emptyset)$ by $e_1 \circ \mathsf{all} \circ e_2 \neq \emptyset$. If $F \subseteq \{^+\}$ and S is a singleton $\{R\}$, it is easy to see that for every expression e in this language there exists a natural number k such that $e \neq \emptyset$ is equivalent to $R^k \neq \emptyset$. The conjunction of $e_1 \neq \emptyset$ and $e_2 \neq \emptyset$ is then expressed using the maximum of the two numbers.

For the only-if direction, first assume $\overline{F}$ does not have all and S contains at least two relation names, say R and T. Now by the Additivity Lemma, the boolean query $R \neq \emptyset \wedge T \neq \emptyset$ is not in $\text{NoAll}^{\neq\emptyset}$.

The other possibility is that $\overline{F}$ does not have all and $F \not\subseteq \{^+\}$. Then $\overline{F}$ must contain at least one of the features converse, projection, or intersection. The case with intersection is proven by Lemma 6 and the case with converse and projection are covered by Lemma 7.

Lemma 6 *For every binary relation name $R \in S$, the boolean query $R^2 \cap R \neq \emptyset \wedge R^3 \cap R \neq \emptyset$ is not in $\text{NoAll}^{\neq\emptyset}$.*

Proof Denote the boolean query $R^2 \cap R \neq \emptyset \wedge R^3 \cap R \neq \emptyset$ by q, and suppose q belongs to $\text{NoAll}^{\neq\emptyset}$ as $e \neq \emptyset$. Let G be the left and H be the right graph in Fig. 6. Since $q(G) = q(H) = \mathit{false}$ and $q(G \cup H) = \mathit{true}$, we have $e(G) = \emptyset$, $e(H) = \emptyset$ and $e(G \cup H) \neq \emptyset$. By the Additivity Lemma, however, $e(G \cup H) = e(G) \cup e(H) = \emptyset$, a contradiction.

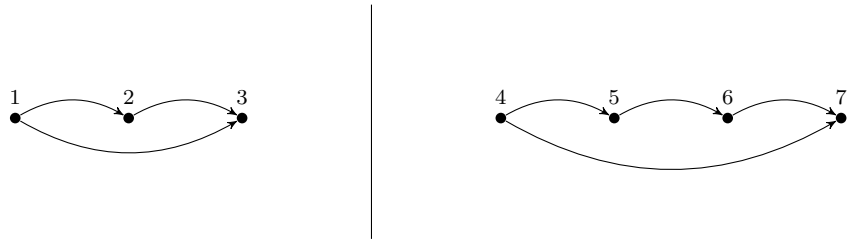

Fig. 6 Graphs used to prove Lemma 6 and Theorem 9.

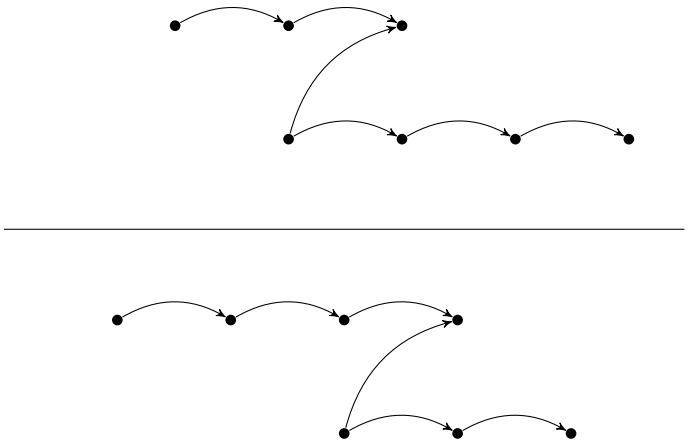

Fig. 7 Graphs used for the proof of Lemma 7.

Using converse, we show that $R^2 \circ R^{-1} \circ R^3 \neq \emptyset \wedge R^3 \circ R^{-1} \circ R^2 \neq \emptyset$ is not in $\text{NoAll}^{\neq\emptyset}$. This result also covers the case with projection. Indeed, both conjuncts are in $\{^{-1}\}^{\neq\emptyset}$, which is subsumed by $\{\pi\}^{\neq\emptyset}$ [16]. Hence, the Lemma 7 also gives a conjunction of $\{\pi\}^{\neq\emptyset}$ queries that is not in $\text{NoAll}^{\neq\emptyset}$. We first show the following short Lemma.

Lemma 7 *For every binary relation name $R \in S$, the boolean query $R^2 \circ R^{-1} \circ R^3 \neq \emptyset \wedge R^3 \circ R^{-1} \circ R^2 \neq \emptyset$ is not in* $\text{NoAll}^{\neq\emptyset}$.

Before we can prove this we need the following technical lemma.

Lemma 8 ([17]) *There is no homomorphism from G_1 to G_2 and vice versa, where G_1 and G_2 are the top and bottom graphs of Fig. 7.*

The lemma follows from the fact that different directed paths of the same length are cores which are not comparable with respect to homomorphisms [17].

Proof (of Lemma 7) Denote the boolean queries $R^2 \circ R^{-1} \circ R^3 \neq \emptyset$ and $R^3 \circ R^{-1} \circ R^2 \neq \emptyset$ by q_1 and q_2 respectively. Consider the graphs G_1 and G_2 shown at the top and bottom of Fig. 7. For every graph G, we have $q_1(G) = \mathit{true}$ iff there is a homomorphism $G_1 \to G$, and similarly for q_2 and G_2. Hence, $q_1(G_2) = \mathit{false}$ and $q_2(G_1) = \mathit{false}$ by Lemma 8.

On the other hand, $q_1 \wedge q_2(G_1 \cup G_2) = \mathit{true}$. Now suppose that $q_1 \wedge q_2$ is expressed by $e \neq \emptyset \in \mathrm{NoAll}^{\neq\emptyset}$. Then, $e(G_1) = e(G_1) = \emptyset$ and $e(G_1 \cup G_2) \neq \emptyset$. By the Additivity Lemma, however, $e(G_1 \cup G_2) = e(G_1) \cup e(G_2) = \emptyset$, a contradiction. □

Turning to the containment modality, we can only offer the general observation that $F^{\subseteq}$ is closed under conjunction whenever F has set difference. Indeed, we can express $e_1 \subseteq e_2 \wedge e_3 \subseteq e_4$ as $(e_1 - e_2) \cup (e_3 - e_4) \subseteq \emptyset$.

At this point we have not been able to prove the converse direction, although we conjecture that set difference in F is indeed necessary for $F^{\subseteq}$ to be closed under conjunction. We have proven two partial results that contribute to the conjecture. Before we can prove them, we need the following two results.

Lemma 9 ([27]) *On the class of complete graphs with at least 3 nodes, every expression in $\mathcal{N}(\mathsf{di}, {}^{-1}, -)$ is equivalent to $\emptyset$, id, di or all.*

Lemma 10 ([27]) *Let e be an expression in $\mathcal{N}(\mathsf{di}, {}^{-1})$.*

1. *If $e(K_3) = \emptyset$ then $e \equiv \emptyset$.*
2. *If $e(K_3) = \mathsf{di}(K_3)$ then $e \equiv \mathsf{di}$.*
3. *If $e(K_3) = \mathsf{id}(K_3)$ then $e \equiv \mathsf{id}$.*

We are now ready for the partial results concerning the failure of closure under conjunction for the containment modality.

Proposition 6 *Let R be a relation name. The boolean query $R^3 \subseteq \mathsf{id} \wedge R^2 \subseteq R$ is not in $\{\mathsf{di}, {}^{-1}, {}^{+}\}^{\subseteq}$.*

Proof Let Q be the boolean query $R^3 \subseteq \mathsf{id} \wedge R^2 \subseteq R$, id_1 be the graph that consists of a single self-loop and $c_2 = \{R(1,2), R(2,3)\}$. Suppose for the sake of contradiction that $e_1 \subseteq e_2 \in \{\mathsf{di}, {}^{-1}, {}^{+}\}^{\subseteq}$ expresses Q. Then, $e_2(K_3)$ equals $\mathsf{all}(K_3)$, $\mathsf{di}(K_3)$, $\mathsf{id}(K_3)$ or $\emptyset(K_3)$ by Lemma 9. In the remainder of the proof we will only work on the graphs K_3, c_2, c_2^+ and id_2, whence we can replace ${}^+$ with unions of compositions. We will now cover each of these scenarios and obtain a contradiction.

If $e_2(K_3) = \mathsf{all}(K_3)$, then $e_1(K_3) \subseteq e_2(K_3)$. We have thus obtained a contradiction, since $Q(K_3) = \mathit{false}$.

If $e_2(K_3) = \mathsf{id}(K_3)$, then $e_2 \equiv \mathsf{id}$ by Lemma 10. Since $Q(c_2) = \mathit{false}$, $e_1(c_2) \not\subseteq e_2(c_2)$, whence we have $e_1(c_2) \cap \mathsf{di}(c_2) \neq \emptyset$. Therefore, $e_1(c_2^+) \cap \mathsf{di}(c_2^+) \neq \emptyset$. We have thus obtained a contradiction, since $Q(c_2^+) = \mathit{true}$.

The case where $e_2(K_3) = \mathsf{di}(K_3)$ is analogous.

If $e_2(K_3) = \emptyset$, then $e_2 \equiv \emptyset$ Lemma 10. Clearly, when $e_1 \not\equiv \emptyset$, then $e_1(\mathsf{id}_2) \neq \emptyset$. We have thus obtained a contradiction, since $Q(\mathsf{id}_2) = \mathit{true}$. □

Unfortunately, we cannot generalize this result to include coprojection. This is because every query $e_1 \subseteq e_2 \wedge e_3 \subseteq \mathsf{id}$, such that $e_1(G) \neq \emptyset$ if $e_3(G) \not\subseteq \mathsf{id}(G)$, does not work to establish separation. Indeed, then $e_1 \subseteq e_2 \wedge e_3 \subseteq \mathsf{id}$ is equivalent with $e_1 \subseteq e_2 \circ \overline{\pi}_1(\mathit{all} \circ (e_3 \cap \mathsf{di}))$.

Next we look at another subfragment of $\mathrm{NoDiff}^{\subseteq}$ that is not closed under conjunction.

Proposition 7 *Let R be a relation name. The boolean query $R^3 \subseteq \emptyset \wedge R^2 \subseteq R$ is not in $\{\cap, \mathsf{id}, \pi, {}^{-1}, {}^{+}\}^{\subseteq}$.*

Proof Let Q be the boolean query $R^3 \subseteq \emptyset \wedge R^2 \subseteq R$, id_1 be the graph that consists of a single self-loop and $c_2 = \{R(1,2), R(2,3)\}$. Suppose for the sake of contradiction that $e_1 \subseteq e_2 \in \{\cap, \mathsf{id}, \pi, {}^{-1}, {}^{+}\}^{\subseteq}$ expresses Q. In the remainder of the proof we will only work on the graphs c_2, c_2^+ and id_1, whence we can replace $^+$ with unions of compositions. Remember that $e_i(\mathsf{id}_1) = R(\mathsf{id}_1)$ unless $e_i \equiv \emptyset$ for $i = 1, 2$. Therefore, if $e_2 \not\equiv \emptyset$ then $e_1(\mathsf{id}_1) \subseteq e_2(\mathsf{id}_1)$. We may thus conclude that $e_2 \equiv \emptyset$. Furthermore, if $e_1(c_2) = \emptyset$, then $e_1 \subseteq e_2$ does not express Q since $Q(c_2) = \mathit{false}$. On the other hand, if $e_1(c_2) \neq \emptyset$, then $e_1(c_2^+) \neq \emptyset$ since the language is monotone. Hence, $e_1 \subseteq e_2$ does not express Q since $Q(c_2^+) = \mathit{true}$. □

Unfortunately, we cannot include coprojection here either. Every query of the form $e_1 \subseteq e_2 \wedge e_3 \subseteq \emptyset$, such that $e_1(G) \neq \emptyset$ if $e_3(G) = \emptyset$, does not work to establish separation. Indeed, then $e_1 \subseteq e_2 \wedge e_3 \subseteq \emptyset$ is equivalent with $e_1 \subseteq e_2 \circ \overline{\pi}_1(\mathsf{all} \circ e_3)$.

6.2.2 Conjunctive queries.

Under nonemptiness, both CQ and UCQ are closed under conjunction, using the same construction as the one used to express tests (proof of Theorem 2).

Under emptiness, note that a family of emptiness queries is closed under conjunction if and only if the corresponding family of nonemptiness queries is closed under *disjunction*. This is clearly the case for UCQ nonemptiness queries. For CQs this happens only rarely:

Theorem 8 *Let S be a database schema. Then, $\mathrm{CQ}_S^{\neq\emptyset}$ is closed under disjunction if and only if S only contains at most two unary relations and no other n-ary relation names with $n \geq 2$.*

Proof First, assume that S only contains unary relations, say $U_1, \ldots, U_n$. Then, boolean queries in $\mathrm{CQ}_S{}^{\neq\emptyset}$ are equivalent to finite conjunctions of queries that test whether the intersection $\bigcap_{U \in A} U$ is nonempty for some $A \subseteq S$. Thus, if S only contains two unary relations, say U_1 and U_2, then $\mathrm{CQ}_S{}^{\neq\emptyset}$ only contains four boolean queries.

- U_1 and U_2 are both nonempty;
- U_1 is nonempty;
- U_2 is nonempty;
- $U_1 \cap U_2$ is nonempty;

Now consider $Q : Q_1 \neq \emptyset \vee Q_2 \neq \emptyset$ where Q_1 and Q_2 are both conjunctive queries over U_1 and/or U_2. Then Q is equivalent to one of the following:

- U_1 and U_2 are both nonempty;
- U_1 is nonempty;
- U_2 is nonempty;
- $U_1 \cap U_2$ is nonempty;
- U_1 or U_2 is nonempty.

The first four queries are respectively expressed by $() \leftarrow U_1(x), U_2(y)$, $() \leftarrow U_1(x)$, $() \leftarrow U_2(x)$ and $() \leftarrow U_1(x), U_2(x)$. The last query, on the other hand, is equivalent to the constant true query since the empty instance is not allowed and there are only two relation names. We may thus conclude that Q is also in $\mathrm{CQ}_S^{\neq\emptyset}$ as desired.

On the other hand, if S contains at least three unary relations, say U_1, U_2 and U_3, then we can consider the CQs $Q_1 = () \leftarrow U_1(x), U_2(x)$ and $Q_2 = () \leftarrow U_3(x)$. Clearly, $Q_1 \vee Q_2 \neq \emptyset$ cannot be expressed by the conjunction of intersection tests.

Finally, suppose that S contains an $n >$ 1-ary relation name. The queries $Q_1 = () \leftarrow R^3 \circ R^{-1} \circ R^2$ and $Q_2 = () \leftarrow R^2 \circ R^{-1} \circ R^3$ are isomorphic to a query over S since we can transform them by replacing $R(x, y)$ with $R(z, \ldots, z, x, y)$. So we may assume that Q_1 and Q_2 are expressible in CQ_S. Suppose now that $Q_1 \neq \emptyset \vee Q_2 \neq \emptyset$ is expressible by a nonemptiness $Q \neq \emptyset$ in $\mathrm{CQ}_S^{\neq\emptyset}$. Notice that $Q_1 \neq \emptyset \vee Q_2 \neq \emptyset \equiv Q_1 \cup Q_2 \neq \emptyset$, whence it is a UCQ_S. Since $Q \sqsubseteq Q_1 \cup Q_2$, we have by the Sagiv–Yannakakis theorem [24] that, either $Q \sqsubseteq Q_1$ or $Q \sqsubseteq Q_2$. This, however, implies that $Q_1(B_{Q_2}) \neq \emptyset$ or $Q_2(B_{Q_1}) \neq \emptyset$ (Recall that B_Q denotes the body of Q). Hence, there is a homomorphism from B_{Q_1} to B_{Q_2} or vice versa. This contradicts Lemma 8 since B_{Q_1} and B_{Q_2} are isomorphic to the top and bottom graphs in Fig. 7 respectively. □

When S contains a binary relation, the result already follows from technical considerations regarding principal filters in the lattice of cores [17]. Indeed, a boolean CQ is a principal filter and the disjunction of two boolean CQs corresponds to the union of two principal filters. The result then follows from the fact that the union of two principal filters of incomparable cores is not principal.

Finally, we consider CQs under containment. Here closure under conjunction happens only in the most trivial setting. We first establish the following lemma.

Lemma 11 *There is no homomorphism from G_1 to G_2 and vice versa, where G_1 and G_2 are the left and right graphs of Fig. 6.*

Proof There cannot be homomorphism from G_2 to G_1 since there is a path of length 3 in G_2 but not in G_1.

Suppose for the sake of contradiction that there is a homomorphism $h : G_1 \to G_2$. Then h has to map 1 to 4 or 1 to 5 since only in 4 and 5 there start paths of length 2. In the former, 3 has to be mapped to 6 and in the latter 3 has to be mapped to 7. However, $(4, 6)$ and $(5, 7)$ are not in G_2. So such a homomorphism cannot exist. □

Theorem 9 *Let S be a database schema. Then, $\mathrm{CQ}_S^{\subseteq}$ is closed under conjunction if and only if S only contains one unary relation and no other n-ary relation names with $n \geq 2$.*

Proof First, suppose that $S = \{U\}$ where U is unary. We show that in this case $\mathrm{CQ}_S{}^{\subseteq}$ only contains two boolean queries:

1. Q_1 : true
2. $Q_2 : (x, y) \leftarrow U(x), U(y) \subseteq (x, x) \leftarrow U(x)$.

Suppose that $e_1 \subseteq e_2 \in \mathrm{CQ}_S{}^{\subseteq}$ where e_1 and e_2 have heads $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ respectively. If $e_1 \subseteq e_2$ is not the constant true boolean query, there exists an instance A such that $e_1(A) \not\subseteq e_2(A)$. Then, there exists i, j such that $x_i \neq x_j$ and $y_i = y_j$ since $(a, \ldots, a) \in Q(I)$ for any CQ over S, any instance I over S, and any $a \in \mathrm{adom}(I)$. Therefore, for any instance I with at least two elements in $\mathrm{adom}(I)$, $e_1 \not\subseteq e_2(I)$. Thus, $e_1 \subseteq e_2$ is equivalent to Q. We may thus conclude that $\mathrm{CQ}_S{}^{\subseteq}$ is closed under conjunction.

On the other hand, suppose that S contains at least two unary relations U_1 and U_2. We now show that $Q : U_1 \subseteq U_2 \wedge U_2 \subseteq U_1$ is not in $\mathrm{CQ}_S{}^{\subseteq}$. Suppose for the sake of contradiction that Q is expressed by $e_1 \subseteq e_2$ in $\mathrm{CQ}_S{}^{\subseteq}$. Let $I_1 = \{U_1(1)\}$ and $I_2 = \{U_2(2)\}$. Clearly, $Q(I_1) = Q(I_2) = \mathit{false}$, whence we have $e_1(I_1) \neq \emptyset$ and $e_1(I_2) \neq \emptyset$. Thus, there is a homomorphism from B_{e_1} into I_1 and B_{e_1} into I_2 (Recall that B_e denotes the body of e). Therefore, $B_{e_1} = \emptyset$. Then, due to safety of CQs, the head of e_1 is empty. Hence, Q is equivalent to $e_2 \neq \emptyset$, which is monotone. This, however, is a contradiction since $U_1 = U_2$ is not monotone.

Finally, suppose that S contains at least one nonunary relation R. Define

- Q_1 as $(x, y) \leftarrow R(x, z, \_, \ldots, \_), R(z, y, \_, \ldots, \_), R(x, y, \_, \ldots, \_)$
- Q_2 as $(x, y) \leftarrow R(x, z_1, \_, \ldots, \_), R(z_1, z_2, \_, \ldots, \_), R(z_2, y, \_, \ldots, \_),$
  $R(x, y, \_, \ldots, \_)$

We now show that $Q : Q_1 \subseteq Q_2 \wedge Q_2 \subseteq Q_1$ is not in $\mathrm{CQ}_S{}^{\subseteq}$. Let G_1 and G_2 be the left and right graphs in Fig. 6 respectively. Clearly, we can identify B_{Q_1} with G_1 and B_{Q_2} with G_2. Suppose for the sake of contradiction that $e_1 \subseteq e_2 \in \mathrm{CQ}_S{}^{\subseteq}$ expresses Q. We first show that there is no homomorphism from G_1 into B_{e_1}. Suppose there is a homomorphism h from G_1 into B_{e_1}. Clearly, $e_1(G_2) \neq \emptyset$ since $Q(G_2) = \mathit{false}$. Hence, there is a homomorphism f from B_{e_1} into G_2. Then, $f \circ h$ is a homomorphism from G_1 to G_2, which contradicts Lemma 11. Analogously, we can establish that there is no homomorphism from G_2 into B_{e_1}.

Since there is no homomorphism from G_1 and G_2 into B_{e_1}, $Q_1(B_{e_1}) = \emptyset$ and $Q_2(B_{e_1}) = \emptyset$, whence we have $Q(B_{e_1}) = \mathit{true}$. Thus, also $e_1(B_{e_1}) \subseteq e_2(B_{e_1})$. Since B_{e_1} is the body of e_1, we have that $e_1 \sqsubseteq e_2$. This is a contradiction since Q is not the constant true query. □

The question whether UCQs under the containment modality are closed under conjunction is still open.

7 Discussion and conclusion

In this paper we have focused on three natural modalities: emptiness, nonemptiness and the containment modalities. Obviously, these are not the only modalities we can consider. For example, we can consider the *equality* modality, i.e., boolean queries of the form $e_1 = e_2$. Notice that the closure under conjunction of the containment modality subsumes the *equality* modality $q_1 = q_2$, which is equivalent to $q_1 \subseteq q_2 \wedge q_2 \subseteq q_1$, as well as to $q_1 \cup q_2 \subseteq q_1 \cap q_2$. Conversely, equality always subsumes containment for any family closed under union, since $q_1 \subseteq q_2$ if and only if $q_1 \cup q_2 = q_2$.

Barwise and Cooper considered the boolean query of the form $R \cap S \neq \emptyset$ for two relations R and S. This query can be stated as "some tuple in R belongs to S". Correspondingly, the modality $e_1 \cap e_2 \neq \emptyset$ states the language construct "some e_1 are e_2". Obviously, most query languages are closed under intersection, so that this modality is subsumed by the nonemptiness modality. But again one may investigate whether the presence of intersection is actually necessary.

More generally, it becomes clear that there is an infinitude of modalities one may consider. A general definition of what constitutes a boolean-query modality may be found in the formal notion of *generalized quantifier* [7,5,18]. The affinity of

generalized quantifiers to natural language constructs makes them interesting as query language constructs.

Obviously, the value of singling out certain boolean query modalities for investigation in a study such as ours will depend on their naturalness as query language constructs. We believe that (non)emptiness and containment are among the most fundamental modalities. It would be too large of a project to provide a complete picture for all relevant boolean query families. Our goal in this paper has been to provide a framework that helps to investigate such matters. We hope we have also provided some interesting results that fit into this framework.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Ameloot, T., Ketsman, B., Neven, F., Zinn, D.: Weaker forms of monotonicity for declarative networking: A more fine-grained answer to the CALM-conjecture. ACM Transactions on Database Systems **40**(4), article 21 (2016)
3. Angles, R., Barceló, P., Rios, G.: A practical query language for graph DBs. In: L. Bravo, M. Lenzerini (eds.) Proceedings 7th Alberto Mendelzon International Workshop on Foundations of Data Management, *CEUR Workshop Proceedings*, vol. 1087 (2013)
4. Angles, R., Gutierrez, C.: Survey of graph database models. ACM Computing Surveys **40**(1), article 1 (2008)
5. Badia, A.: Quantifiers in Action: Generalized Quantification in Query, Logical and Natural Languages, *Advances in Database Systems*, vol. 37. Springer (2009)
6. Barceló, P.: Querying graph databases. In: Proceedings 32st ACM Symposium on Principles of Databases, pp. 175–188. ACM (2013)
7. Barwise, J., Cooper, R.: Generalized quantifiers and natural language. Linguistics and Philosophy **4**(2), 159–219 (1981)
8. Beeri, C., Vardi, M.: A proof procedure for data dependencies. J. ACM **31**(4), 718–741 (1984)
9. Van den Bussche, J.: Applications of Alfred Tarski's ideas in database theory. In: L. Fribourg (ed.) Computer Science Logic, *Lecture Notes in Computer Science*, vol. 2142. Springer (2001)
10. ten Cate, B., Marx, M.: Navigational XPath: Calculus and algebra. SIGMOD Record **36**(2), 19–26 (2007)
11. Chandra, A., Merlin, P.: Optimal implementation of conjunctive queries in relational data bases. In: Proceedings 9th ACM Symposium on the Theory of Computing, pp. 77–90. ACM (1977)
12. Cruz, I., Mendelzon, A., Wood, P.: A graphical query language supporting recursion. In: U. Dayal, I. Traiger (eds.) Proceedings of the ACM SIGMOD 1987 Annual Conference, *SIGMOD Record*, vol. 16:3, pp. 323–330. ACM Press (1987)
13. Ebbinghaus, H.D., Flum, J.: Finite Model Theory, second edn. Springer (1999)
14. Fletcher, G., Gyssens, M., Leinders, D., Van den Bussche, J., Van Gucht, D., Vansummeren, S., Wu, Y.: Relative expressive power of navigational querying on graphs. In: Proceedings 14th International Conference on Database Theory (2011)
15. Fletcher, G., Gyssens, M., Leinders, D., Van den Bussche, J., Van Gucht, D., Vansummeren, S., Wu, Y.: The impact of transitive closure on the expressiveness of navigational query languages on unlabeled graphs. Annals of Mathematics and Artificial Intelligence **73**(1–2), 167–203 (2015)
16. Fletcher, G., Gyssens, M., Leinders, D., Surinx, D., Van den Bussche, J., Van Gucht, D., Vansummeren, S., Wu, Y.: Relative expressive power of navigational querying on graphs. Information Sciences **298**, 390–406 (2015)
17. Hell, P., Nesetril, J.: Graphs and Homomorphisms. Oxford Lecture Series in Mathematics and Its Applications. OUP Oxford (2004). URL `https://books.google.be/books?id=bJXWV-qK7kYC`
18. Hella, L., Luosto, K., Väänänen, J.: The hierarchy theorem for generalized quantifiers. The Journal of Symbolic Logic **61**(3), 802–817 (1996)

19. Imielinski, T., Lipski, W.: The relational model of data and cylindric algebras. J. Comput. Syst. Sci. **28**, 80–102 (1984)
20. Kolaitis, P.: On the expressive power of logics on finite models. In: Finite Model Theory and Its Applications, chap. 2. Springer (2007)
21. Libkin, L.: Elements of Finite Model Theory. Springer (2004)
22. Libkin, L., Martens, W., Vrgoč, D.: Quering graph databases with XPath. In: Proceedings 16th International Conference on Database Theory. ACM (2013)
23. Marx, M., de Rijke, M.: Semantic characterizations of navigational XPath. SIGMOD Record **34**(2), 41–46 (2005)
24. Sagiv, Y., Yannakakis, M.: Equivalences among relational expressions with the union and difference operators. J. ACM **27**(4), 633–655 (1980)
25. Surinx, D.: A framework for comparing query languages in their ability to express boolean queries. Ph.D. thesis, Hasselt University (2017). `http://dsurinx.be/phd.pdf`
26. Surinx, D., Van den Bussche, J.: A monotone preservation result for boolean queries expressed as a containment of conjunctive queries (2018). URL `http://arxiv.org/abs/1808.08822`
27. Surinx, D., Van den Bussche, J., Van Gucht, D.: The primitivity of operators in the algebra of binary relations under conjunctions of containments. In: Proceedings 32nd Annual ACM/IEEE Symposium on Logic in Computer Science. IEEE Computer Society Press (2017)
28. Surinx, D., Van den Bussche, J., Van Gucht, D.: A framework for comparing query languages in their ability to express boolean queries. In: F. Ferrarotti, S. Woltran (eds.) Proceedings of the 10th International Symposium on Foundations of Information and Knowledge Systems, *Lecture Notes in Computer Science*, vol. 10833, pp. 360–378. Springer (2018)
29. Surinx, D., Fletcher, G., Gyssens, M., Leinders, D., Van den Bussche, J., Van Gucht, D., Vansummeren, S., Wu, Y.: Relative expressive power of navigational querying on graphs using transitive closure. Logic Journal of the IGPL **23**(5), 759–788 (2015)
30. Wood, P.: Query languages for graph databases. SIGMOD Record **41**(1), 50–60 (2012)