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

Dimitri Surinx[1], Jan Van den Bussche[1], and Dirk Van Gucht[2]

[1] Hasselt University, Belgium
[2] Indiana University, Bloomington, IN, USA

Abstract. We identify three basic modalities for expressing boolean queries using the expressions of a query language: nonemptiness, emptiness, and containment. For the class of first-order queries, these three 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. The expressive power under these different modalities can be compared in several different ways, e.g., we can compare a fixed query language $\mathcal{F}$ under emptiness to $\mathcal{F}$ under nonemptiness.
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. As an important auxiliary result, we establish a preservation theorem for monotone containments of conjunctive queries.

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 [1–4], 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 $e_2(D)$.[3] 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, 6] can be regarded as the containment of two conjunctive queries. Similarly, an equality-generating dependency 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 depending 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.

[3] 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 [5, 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.

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 [7–14]. 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 offer a general preservation theorem for monotone containments of CQs: these boolean queries are exactly the nonemptiness CQs. Using this result, and earlier results on the nonexpressibility of certain nonemptiness queries [15], we can 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.

Several proofs have been omitted due to space limitations, and will be included in a following journal publication.

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, 17, 15] and under the containment modality [18]. The present paper is complementary in that it emphasizes the comparison of different boolean query modalities, for fixed fragments or across fragments.

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, relational algebra, conjunctive queries [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, 20]. 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:

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

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 to $\mathrm{adom}(I)$ if $I(R) \not\subseteq I(S)$ 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 results in this paper 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, 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.

The most basic language we consider for expressing binary queries over graphs 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 [18].

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 [10–14].

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.[4]

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.

Various interdependencies exist between the nonbasic features [12]:

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

[4] 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 [21].

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.

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 nonbasic 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) $\mathcal{F}$. Formally, 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 $\neg true = false$ and $\neg false = true$. 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 $q \subseteq \text{empty}$. □

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 in $\{\emptyset, R, S, R \cup S\}$. 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* $\mathrm{MON}^{=\emptyset} \cap \mathrm{MON}^{\neq\emptyset}$ *are the constant* true *and* false *queries.*

As a corollary, 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 $\mathcal{F}^{\subseteq} \not\subseteq \mathcal{F}^{=\emptyset}$.*

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 [22]. 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.

Both cylindrification and tests run counter to additivity. For example, just computing $\text{adom}(I) \times \text{adom}(I)$ 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* $\text{ADD}^{\neq\emptyset} \cap \text{ADD}^{\subseteq}$ *are the constant* true *and* false *queries.*

As a 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 $\mathcal{F}^{=\emptyset} \not\subseteq \mathcal{F}^{\neq\emptyset}$.*

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.

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.

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 1. 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 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 is proven by showing that $\mathcal{N}(F)$ fulfills the conditions of Theorem 1.

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

For the first part of the theorem, 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 $\mathrm{NoDiff}^{=\emptyset}$.

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* $\mathrm{NoDiff}^{=\emptyset}$ *nor in* $\mathrm{NoDiff}^{\neq\emptyset}$.

The only-if directions of the remaining parts of the theorem all revolve around the fragment $\mathrm{NoAll} = \{{}^{-1}, -, {}^{+}\}$, 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. Hence we obtain the following:

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

This lemma can be proven directly but also follows from the additivity of connected stratified Datalog [22].

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) [23]. Theorem 3 directly tells us that $\text{RPQ}^{=\emptyset} \not\subseteq \text{RPQ}^{\neq\emptyset}$, $\text{RPQ}^{\subseteq} \not\subseteq \text{RPQ}^{=\emptyset}$ and $\text{RPQ}^{\subseteq} \not\subseteq \text{RPQ}^{\neq\emptyset}$.

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 [15, 18].

The next step, then, is to see how different query languages relate when using different modalities. In this paper, we investigate how $F_1^{\neq\emptyset}$ compares to $F_2^{\subseteq}$, for different navigational graph query language fragments F_1 and F_2. This question is interesting especially 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 to try to understand to what extent the containment modality, using some language F_2, can be used to express nonemptiness queries using some other language F_1.

Example 4. For a positive example, the query $R^2 \circ R^{-1} \circ R^2 \neq \emptyset$ in $\{^{-1}\}^{\neq\emptyset}$ is expressed by $\mathsf{all} \subseteq \mathsf{all} \circ \pi_1(R^2 \circ \pi_2(\pi_1(R) \circ R)) \circ \mathsf{all}$ 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 $\{\mathsf{all}\}^{\subseteq}$.

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 [15]. We refer to the paper [15], which shows that $F_1^{\neq\emptyset} \not\subseteq F_2^{\neq\emptyset}$ can only happen in the following cases:

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

Transitive closure: Transitive closure is 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 $F_2 \subseteq \{\mathsf{all}, \mathsf{di}, ^+\}$.

Projection: Projection 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.

Intersection. The largest fragment for F_2 we need to consider is NoInt $= \{\mathsf{di}, \overline{\pi}, ^{-1}, ^+\}$ ("no intersection"). We can show that the query $R \cap \mathsf{id} \neq \emptyset$ ("the graph has self-loops") is not in NoInt$^{\subseteq}$.

Difference. We can show that $R^2 - R \neq \emptyset$ ("the graph is not transitive") is not in NoDiff$^{\subseteq}$.

Diversity. We can show that $\mathsf{di} \neq \emptyset$ ("the graph has at least two nodes") is not in $\{\mathsf{all}, ^{-1}, \overline{\pi}, \cap, ^+\}^{\subseteq}$.

Coprojection. We can show that $\overline{\pi}_1(R) \neq \emptyset$ ("the graph has at least one sink node") is not in $\{\mathsf{di}, ^{-1}, \cap, ^+\}^{\subseteq}$.

Transitive closure. From our earlier work 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.

Converse. The largest fragment without converse is NoConv $= \{\mathsf{di}, \pi, ^+, -\}$. Since NoConv has both all and $-$, we have NoConv$^{\neq\emptyset}$ = 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 NoConv$^{\neq\emptyset}$. In case (b), we already know [15, Proposition 5.4] that $R^2 \circ (R \circ R^{-1})^+ \circ R^2 \neq \emptyset$ is not in NoConv$^{\neq\emptyset}$. To settle case (c), we can show that $R^2 \circ R^{-1} \circ R^2 \neq \emptyset$ is not in $\{\mathsf{di}, ^+\}^{\subseteq}$.

A preservation result. 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.

Note that such a result would fit the profile of a preservation theorem since it gives a syntactical characterization for a semantical property (here monotonicity). Preservation theorems have been studied intensively in model theory, finite model theory and database theory [24–29].

We can give a partial answer in the form of the following preservation result, which we believe to be interesting in its own right. In the following Theorem, the conjunctive queries need not be safe. (A CQ is safe if all variables in its head are present in its body). This is important for the application to graph queries in the corollary; to express all we need an unsafe CQ.

Theorem 4. *Let Q_1 and Q_2 be conjunctive queries so that the boolean containment query $Q_1 \subseteq Q_2$ is monotone. Then $Q_1 \subseteq Q_2$ is also expressible as a nonemptiness query $P \neq \emptyset$, where P is a conjunctive query. Moreover, the body of P can be taken so that it is part of the body of Q_2.*

Corollary 1. *$\{\pi\}^{\neq\emptyset} \not\subseteq F_2^{\subseteq}$, where F_2 is the union-free fragment of $\{\mathsf{all}, {}^{-1}\}$.*

Proof. Path queries expressed in the union-free fragment of $\mathcal{N}(\mathsf{all}, {}^{-1})$ are expressible as conjunctive queries. As mentioned above, we know there exists a (monotone) boolean query Q in $\{\pi\}^{\neq\emptyset}$ that is not in $\{\mathsf{all}, {}^{-1}\}^{\neq\emptyset}$. If Q would be in $\{\mathsf{all}, {}^{-1}\}^{\subseteq}$, the above Theorem would imply Q in $\{\mathsf{all}, {}^{-1}\}^{\neq\emptyset}$, a contradiction.

It is an interesting challenge to try to extend Theorem 4 to unions of CQs, CQs with nonequalities, and perhaps even recursive (Datalog) programs.

6 Closure under boolean connectives

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. For navigational graph query language fragments F, we have closure under negation of $F^{\subseteq}$ only if both all and $-$ are in $\overline{F}$. When $\mathcal{F}$ is the family of conjunctive queries, or unions of conjunctive queries, $\mathcal{F}^{\subseteq}$ is again not closed under negation. We omit the details.

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.

We begin this investigation for our navigational graph query language fragments. 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 5. *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 $\{R\}$, the language $\mathcal{N}(F)$ is very simple and $F^{\neq\emptyset}$ is easily seen to be closed under conjunction.

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 $\mathrm{NoAll}^{\neq\emptyset}$.

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. Using intersection, we can show that $R^2 \cap R \neq \emptyset \wedge R^3 \cap R \neq \emptyset$ is not in $\mathrm{NoAll}^{\neq\emptyset}$.

Using converse, we can 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 $\mathrm{NoAll}^{\neq\emptyset}$. 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 also gives a conjunction of $\{\pi\}^{\neq\emptyset}$ queries that is not in $\mathrm{NoAll}^{\neq\emptyset}$.

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. Two partial results we could prove are that $R^3 \subseteq \mathsf{id} \wedge R^2 \subseteq R$ is not in $\{\mathsf{di}, ^{-1}, ^+\}^{\subseteq}$, and that $R^3 \subseteq \emptyset \wedge R^2 \subseteq R$ is not in $\{\cap, \pi, ^{-1}, ^+\}^{\subseteq}$. The difficulty here is to extend this allowing coprojection.

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 6. *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 $n \geq 2$.*

Finally, we consider CQs under containment. Here closure under conjunction happens only in the most trivial setting.

Theorem 7. *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 $n \geq 2$.*

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

7 Discussion and conclusion

Observe 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 $q_1 \cup q_2 = q_2$.

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* [30, 31]. The affinity of generalized quantifiers to natural language constructs makes them interesting as query language constructs. For example, for two relations R and S, Barwise and Cooper consider the boolean query $R \cap S \neq \emptyset$. 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.

Questions of the same nature as the ones studied here have also been studied by logicians interested in generalized quantifiers. For example, Hella et al. [32] showed that for every finite set of generalized quantifier there is a more powerful one (by moving to more or higher-arity relations).

Obviously, the value of singling out certain generalized quantifiers 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. Ebbinghaus, H.D., Flum, J.: Finite Model Theory. second edn. Springer (1999)
3. Libkin, L.: Elements of Finite Model Theory. Springer (2004)
4. Kolaitis, P.: On the expressive power of logics on finite models. In: Finite Model Theory and Its Applications. Springer (2007)
5. Chandra, A., Merlin, P.: Optimal implementation of conjunctive queries in relational data bases. In: Proceedings 9th ACM Symposium on the Theory of Computing, ACM (1977) 77–90
6. Beeri, C., Vardi, M.: A proof procedure for data dependencies. Journal of the ACM **31**(4) (1984) 718–741
7. Angles, R., Gutierrez, C.: Survey of graph database models. ACM Computing Surveys **40**(1) (2008) article 1
8. Wood, P.: Query languages for graph databases. SIGMOD Record **41**(1) (March 2012) 50–60

9. Barceló, P.: Querying graph databases. In: Proceedings 32st ACM Symposium on Principles of Databases, ACM (2013) 175–188
10. Marx, M., de Rijke, M.: Semantic characterizations of navigational XPath. SIGMOD Record **34**(2) (2005) 41–46
11. ten Cate, B., Marx, M.: Navigational XPath: Calculus and algebra. SIGMOD Record **36**(2) (2007) 19–26
12. 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)
13. Libkin, L., Martens, W., Vrgoč, D.: Quering graph databases with XPath. In: Proceedings 16th International Conference on Database Theory, ACM (2013)
14. Angles, R., Barceló, P., Rios, G.: A practical query language for graph DBs. In Bravo, L., Lenzerini, M., eds.: Proceedings 7th Alberto Mendelzon International Workshop on Foundations of Data Management. Volume 1087 of CEUR Workshop Proceedings. (2013)
15. 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) (2015) 759–788
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** (2015) 390–406
17. 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) (2015) 167–203
18. 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)
19. Imielinski, T., Lipski, W.: The relational model of data and cylindric algebras. Journal of Computer and System Sciences **28** (1984) 80–102
20. Van den Bussche, J.: Applications of Alfred Tarski's ideas in database theory. In Fribourg, L., ed.: Computer Science Logic. Volume 2142 of Lecture Notes in Computer Science., Springer (2001)
21. Surinx, D.: A Framework for Comparing Query Languages in Their Ability to Express Boolean Queries. PhD thesis, Hasselt University (2017) `http://dsurinx.be/phd.pdf`.
22. 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) (2016) article 21
23. Cruz, I., Mendelzon, A., Wood, P.: A graphical query language supporting recursion. In Dayal, U., Traiger, I., eds.: Proceedings of the ACM SIGMOD 1987 Annual Conference. Volume 16:3 of SIGMOD Record., ACM Press (1987) 323–330
24. Chang, C., Keisler, H.: Model Theory. 3rd edn. North-Holland (1990)
25. Benedikt, M., Leblay, J., ten Cate, B., Tsamoura, E.: Generating plans from proofs: The interpolation-based approach to query reformulation. Morgan&Claypool (2016)
26. Rossman, B.: Homomorphism preservation theorems. Journal of the ACM **55**(3) (2008) article 15

27. Gurevich, Y.: Toward logic tailored for computational complexity. In Richter, M., et al., eds.: Computation and Proof Theory. Volume 1104 of Lecture Notes in Mathematics. Springer-Verlag (1984) 175–216
28. Ajtai, M., Gurevich, Y.: Monotone versus positive. Journal of the ACM **34**(4) (1987) 1004–1015
29. Stolboushkin, A.: Finitely monotone properties. In: Proceedings 10th Annual IEEE Symposium on Logic in Computer Science. (1995) 324–330
30. Barwise, J., Cooper, R.: Generalized quantifiers and natural language. Linguistics and Philosophy **4**(2) (1981) 159–219
31. Badia, A.: Quantifiers in Action: Generalized Quantification in Query, Logical and Natural Languages. Volume 37 of Advances in Database Systems. Springer (2009)
32. Hella, L., Luosto, K., Väänänen, J.: The hierarchy theorem for generalized quantifiers. The Journal of Symbolic Logic **61**(3) (1996) 802–817