DOI: 10.1109/LICS.2017.8005122
Handle: http://hdl.handle.net/1942/25325

The primitivity of operators in the algebra of binary relations under conjunctions of containments

Dimitri Surinx
Hasselt University
dimitri.surinx@uhasselt.be

Jan Van den Bussche
Hasselt University
jan.vandenbussche@uhasselt.be

Dirk Van Gucht
Indiana University
vgucht@cs.indiana.edu

***Abstract*—The algebra of binary relations provides union and composition as basic operators, with the empty set as neutral element for union and the identity relation as neutral element for composition. The basic algebra can be enriched with additional features. We consider the diversity relation, the full relation, intersection, set difference, projection, coprojection, converse, and transitive closure. It is customary to express boolean queries on binary relational structures as finite conjunctions of containments. We investigate which features are primitive in this setting, in the sense that omitting the feature would allow strictly less boolean queries to be expressible. Our main result is that, modulo a finite list of elementary interdependencies among the features, every feature is indeed primitive.**

I. Introduction

The algebra of binary relations (aka the calculus of relations) was created by De Morgan, Peirce, and Schröder, and popularized and further developed by Tarski and his collaborators [1]–[3]. These developments gave rise to the rich field of relation algebras [4]–[6]. In the present paper, however, we are focusing on the algebra of binary relations as a language for expressing properties of binary relational structures. This focus comes very naturally in the context of query languages for graph data. Indeed, a binary relational structure really is a directed graph, with the different binary relation names playing the role of edge labels. Not surprisingly, the algebra of binary relations lies at the basis of query languages for graph databases [7]–[12].

But also more fundamentally, the algebra of binary relations originated in the desire to have a principled language for expressing properties (axiomatizing classes) of relational structures. (In this respect, the algebra of binary relations actually predates first-order logic [13].) This paper is part of our ongoing work [9], [14]–[17] to understand the expressive power, in particular the primitivity or interdependencies, among different operators considered in the realm of binary relational algebra. Since we include projection, coprojection, transitive closure, intersection, and converse, our investigation is also relevant to propositional dynamic logic, multi-dimensional modal logic [18] and to the relational interpretation of Kleene algebras with tests, and of Kleene allegories [19], [20].

There are indeed many different operators that have been considered. Our choice of operators is a natural one and motivated by the applications to graph database query languages. We always start from union $\cup$ and composition $\circ$, as the equivalents of addition and multiplication for binary relations, together with the empty relation $\emptyset$ and the identity relation id, as the neutral elements for union and composition. These four features are always included. Then we consider arbitrary language fragments built up by adding any subset of the following features: the full relation all; the diversity relation di; converse $^{-1}$; transitive closure $^{+}$; intersection $\cap$; set difference $-$; projection π; and coprojection $\overline{\pi}$. These operators are self-explanatory, except perhaps projection and coprojection which provide for testing and negative testing.

The task of understanding and comparing language fragments that include some needed features, but omit unneeded ones, makes sense. For example, in database query processing we could use data structures or query optimization strategies that work well for some operators but not for others. Moreover, some automated reasoning tasks, such as satisfiability or subsumption testing, will be decidable in some fragments but not in the full language.

Expressions are built up from relation names, constants, and operators. Given a graph G appropriate for an expression e, we can evaluate e in G and obtain a binary relation $e(G)$ over the domain of G. Thus, expressions define what we call *path queries*: functions that map graphs to relations. For example, consider the expression $(R \circ R \circ S) \cap id$ applied to a graph with R- and S-labeled edges. Thinking of R as bus rides and of S as train rides, the expression returns all identical pairs (x, x) of locations x from where you can take two consecutive bus rides and get back home by a single train ride.

In logic, however, relational algebra expressions are also often used in equations or containments (or implications). For example, a binary relation R is a total order if and only if it satisfies the four containments $id \subseteq R$; $R \circ R \subseteq R$; $R \cap R^{-1} \subseteq id$; and $all \subseteq R \cup R^{-1}$. The second chapter of Maddux's book [4] is full of such examples. Thus, conjunctions of containments are used to express what we call *boolean queries*: functions mapping graphs to true/false. Another example is the single containment $all \subseteq (R \cup R^{-1})^{+}$ which expresses that the graph is connected. Also in databases, containments are very natural, for example, in the expression of referential integrity constraints. For example, when we think of R as works-for and of S as managed-by, then the containment $\pi_2(S) \subseteq \pi_1(R)$ expresses that every manager is also an employee. As a last example, R is a function iff $R^{-1} \circ R \subseteq id$.

Our main goal in this paper is to investigate the primitivity

978-1-5090-3018-7/17/$31.00

of the operators under conjunctions of containments. For example, take a fragment F that does not include the converse operator, and consider the fragment $F' = F \cup \{^{-1}\}$ that adds converse to F. Can we express strictly more boolean queries (as conjunctions of containments) using F'-expressions than using only F-expressions? Note that for path queries, primitivity is a rather easy exercise. For example, it is not difficult to see that the path query R^{-1} cannot be expressed without using converse. For boolean queries, however, this is often less obvious. In previous work [14]–[16], we considered boolean queries expressed by nonemptiness statements $e \neq \emptyset$. It turns out, for example, that converse is not always primitive in that setting. Specifically, for any fragment F lacking intersection but including projection, all boolean queries expressed by the nonemptiness of an F-expression are already expressible using an F-expression that does not use converse [14]. For instance, $R^3 \circ R^{-1} \circ R^3$ is nonempty if and only if $R^3 \circ \pi_2(\pi_1(R^3) \circ R)$ is nonempty. Furthermore, transitive closure is not primitive for nonemptiness queries over a single relation name in fragments that only add di or π as extra features [16]. We thus see that primitivity in general can depend on the vocabulary, as well as on the fragment under consideration.

In contrast, in this paper, we show that under conjunctions of containments, the interdependencies among the operators are exactly the same as for path queries. One might say that the surprise is that there are no surprises! We already know that the operator interdependencies for path queries can be concisely summarized in the form of a finite list of explicit definitions [9]. (This list contains definitions such as $\pi_1(e) = (e \circ e^{-1}) \cap id$.) So, our result implies that this list of definitions is complete also for determining the interdependencies under conjunctions of containments.

In this work, our creativity went mainly in deliberating for each operator and every fragment whether the operator might be nonprimitive in the fragment. In the end, failing to find any collapses, we needed to find suitable boolean queries that show primitivity for each operator. We subsequently could find quite elementary arguments that the query is indeed inexpressible without using the operator.

II. Definitions and results

Let us fix a nonempty relational vocabulary Λ, i.e., a nonempty set of relation names. All results in this paper hold for every choice of Λ. A *graph* is a structure $G = (V, (R^G)_{R \in \Lambda})$ where V is a nonempty set (the domain of G) and each R^G is a binary relation on V. The domain V is denoted by $\mathrm{dom}(G)$ and its elements are called the nodes of G. Also, each pair in R^G is called an *edge* with label R. Graphs may be infinite, unless explicitly stated otherwise. All inexpressibility results in this paper already hold when restricting to finite graphs, however.

The most basic query language for graphs we consider is the algebra $\mathcal{N}$. The expressions of $\mathcal{N}$ are built recursively from the relation names in Λ, the constant symbol $\emptyset$, and the constant symbol id, using composition $(e_1 \circ e_2)$ and union $(e_1 \cup e_2)$.

$$
\begin{aligned}
R(G) &= R^G \\
\emptyset(G) &= \emptyset \\
id(G) &= \{(m,m) \mid m \in \mathrm{dom}(G)\} \\
e_1 \cup e_2(G) &= e_1(G) \cup e_2(G) \\
e_1 \circ e_2(G) &= \{(m,n) \mid \exists p : (m,p) \in e_1(G) \\
&\qquad \wedge (p,n) \in e_2(G)\}
\end{aligned}
$$

Fig. 1. Semantics of $\mathcal{N}$.

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

Fig. 2. Semantics of the nonbasic features.

Remark II.1. The assumption of a basic language is a point of discussion. In principle there is no reason to use a basic language at all: just consider each and every operation to be optional. For our investigation, we have chosen for a basic language for the following reasons. First, it lends structure to the investigation. Without the framework provided by a basic language, our task would include a large number of ad-hoc cases to be settled. Furthermore, the field of relation algebras identifies composition and union as the natural counterparts for multiplication and addition of binary relations. Union is a very mild operation that is computationally simple. Without composition, you can hardly say you are investigating binary relations. Adding the neutral elements (empty for union, identity for composition) provides the mathematically natural structure of a semiring. □

Semantically, each expression $e \in \mathcal{N}$ defines a *path query*, a function q from graphs to binary relations, mapping a graph G to a binary relation on $\mathrm{dom}(G)$. See Figure 1.

We will use the abbreviation R^n for $R \circ \cdots \circ R$ (n times R, with $n > 0$). Also, R^0 is used as a synonym for id.

The basic algebra $\mathcal{N}$ can be extended by adding some of the following operators: the constant symbols for diversity (di) and for the full relation (all); the unary operators converse (e^{-1}), transitive closure (e^{+}), projection ($\pi_1(e)$ and $\pi_2(e)$), and coprojection ($\overline{\pi}_1(e)$ and $\overline{\pi}_2(e)$); and the binary operators intersection ($e_1 \cap e_2$) and set difference ($e_1 - e_2$).

We refer to the operators in the basic algebra $\mathcal{N}$ as *basic features*; we refer to the extensions as *nonbasic features*. The

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

Fig. 3. Interdependencies among the nonbasic features [9].

semantics of the latter are given in Figure 2. Note that the complement e^c can be expressed as $all - e$.

Two expressions e_1 and e_2 are *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.

A *fragment* is any set F of nonbasic features, where we take either both projections or none of them, and the same for coprojection. Formally, $\pi_1 \in F$ iff $\pi_2 \in F$, and the same for $\overline{\pi}_1$ and $\overline{\pi}_2$. We denote by $\mathcal{N}(F)$ the language obtained by adding the features in F to $\mathcal{N}$. For example, $\mathcal{N}(\cap)$ denotes the extension with intersection, and $\mathcal{N}(\cap, \pi)$ denotes the extension with intersection and both projections.

Various interdependencies exist between the nonbasic features [9]. They are shown in Figure 3. For example, by the third equivalence in the figure, when we add difference, we get intersection for free.

Hence, a feature f may be present in the language $\mathcal{N}(F)$ without belonging to F. To deal with this, we use the *completion* $\overline{F}$ of a set of nonbasic features F. Guided by Figure 3, we define $\overline{F}$ as the smallest superset of F satisfying the following rules:

- if $di \in F$, then $all \in \overline{F}$;
- if $all \in F$ and $- \in F$, then $di \in \overline{F}$;
- if $- \in F$, then $\cap \in \overline{F}$;
- if $\cap \in \overline{F}$ and ($^{-1} \in F$ or $all \in \overline{F}$), then $\pi \in \overline{F}$;
- if $\overline{\pi} \in F$, then $\pi \in \overline{F}$;
- if $- \in F$ and $\pi \in \overline{F}$, then $\overline{\pi} \in \overline{F}$.

For example,

$$\overline{\{all, -\}} = \overline{\{di, -\}} = \{di, all, \cap, -, \pi, \overline{\pi}\}.$$

It is clear that the languages $\mathcal{N}(F)$ and $\mathcal{N}(\overline{F})$ are equivalent in that they can express precisely the same path queries. For any two fragments F_1 and F_2, call $\mathcal{N}(F_1)$ *subsumed* by $\mathcal{N}(F_2)$, denoted by $F_1 \leq F_2$, if every path query expressible in $\mathcal{N}(F_1)$ is also expressible in $\mathcal{N}(F_2)$. The above list of equivalences is known to be complete for determining subsumption for path queries. Specifically, we have $F_1 \leq F_2$ if and only if $F_1 \subseteq \overline{F_2}$ [14].

In the fragment $\mathcal{N}(di, -, ^{-1})$, which is the largest fragment (up to completion) without transitive closure, we can express exactly the same path queries as the 3-variable fragment of first-order logic [2], [18].

A. Boolean queries

A *boolean query* is a function that maps graphs to true/false. Equivalently, a boolean query is a class of graphs. It is very common in algebraic logic to express boolean queries as finite conjunctions of *containment statements*. A containment statement is of the form $e_1 \subseteq e_2$, for expressions e_1 and e_2. It holds in a graph G if $e_1(G) \subseteq e_2(G)$. For any fragment F, the family of boolean queries expressible by finite conjunctions of containment statements using expressions from $\mathcal{N}(F)$ is denoted by $F^{\wedge\subseteq}$.

Our main result is the following:

Theorem II.2. *For any two fragments F_1 and F_2, we have $F_1^{\wedge\subseteq} \subseteq F_2^{\wedge\subseteq}$ if and only if $F_1 \subseteq \overline{F_2}$.*

So, subsumption among fragments under conjunctions of containments behaves the same as subsumption for path queries. This is not obvious. Indeed, as already indicated in the Introduction, in earlier work we have shown that when we express boolean queries as nonemptiness statements $e \neq \emptyset$, subsumption behaves very differently [14]–[16].

We call a nonbasic feature f *primitive* (under conjunctions of containments) if for any fragment F such that $f \notin \overline{F}$, we have $\{f\}^{\wedge\subseteq} \not\subseteq F^{\wedge\subseteq}$. In other words, just the feature, combined with the basic features, is enough to express some boolean query that is not expressible without using the feature. We can then reformulate the above theorem as saying that *every nonbasic feature is primitive*. We next devote one section to every nonbasic feature.

III. Projection

Up to completion, there are two maximal fragments lacking projection: $\{di, ^{-1}, ^{+}\}$ and $\{-, ^{+}\}$. The latter fragment can be quickly dealt with using previous work. Note that the query in the following proposition is an emptiness query, which is a special case of a containment query since $e = \emptyset$ iff $e \subseteq \emptyset$.

Proposition III.1. *Let R be a relation name. The boolean query $\pi_1(R^2) \circ R \circ \pi_2(R^2) = \emptyset$ is not in $\{-, ^{+}\}^{\wedge\subseteq}$.*

Proof. Let e be the expression $\pi_1(R^2) \circ R \circ \pi_2(R^2)$. In previous work [14, Proposition 5.6(2)] we have shown that the boolean query $e \neq \emptyset$ is not expressible as the nonemptiness of an expression in $\mathcal{N}(-, ^{+})$. Equivalently, the boolean query $e = \emptyset$ is not expressible as the emptiness of an expression in $\mathcal{N}(-, ^{+})$. The result now follows from the following obvious proposition. □

Proposition III.2. *Let F be a fragment with set difference. Then every boolean query in $F^{\wedge\subseteq}$ can be expressed as the emptiness of an expression in $\mathcal{N}(F)$.*

Proof. We can express, say, $e_1 \subseteq e_2 \wedge e_3 \subseteq e_4$ as $(e_1 - e_2) \cup (e_3 - e_4) = \emptyset$. □

We now have our hands free to deal with the fragment $\{di, ^{-1}, ^{+}\}$. We will show:

Proposition III.3. *The boolean query $R \circ \pi_1(R) \subseteq id$ is not in $\{di, ^{-1}, ^{+}\}^{\wedge\subseteq}$.*

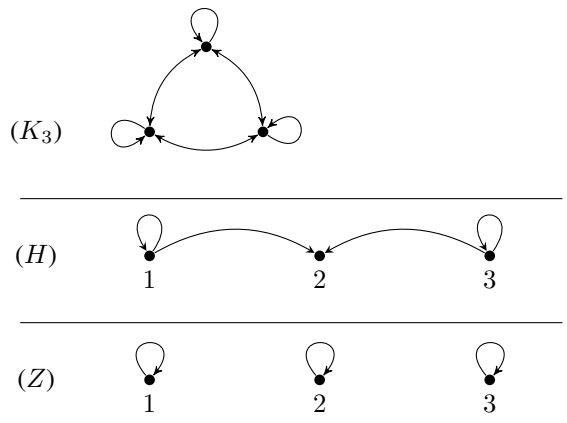

Fig. 4. Graphs used in the proof of Proposition III.3.

To prove this proposition it will suffice to reason on the three finite graphs called K_3, H and Z, shown in Figure 4. The edges in these graphs are all understood to be labeled by the same relation name R. Note that K_3 is the complete graph (with loops) on 3 nodes; in general we use K_n to denote the complete graph on n nodes. On a complete graph, any path query invariant under isomorphisms can return only $\emptyset$, id, di, or all. Given the connection with the 3-variable fragment of first-order logic mentioned earlier, the following lemma is obvious.

Lemma III.4. *On the class of complete graphs with at least 3 nodes, every expression in $\mathcal{N}(di, ^{-1}, -)$ is equivalent to $\emptyset$, id, di or all.*

For expressions in $\mathcal{N}(di, ^{-1})$ in particular, the outcome on K_3 may determine the complete behavior on all graphs, in the sense of the following lemma.

In the proof, and also later in the proof of Proposition IV.1, we appeal to monotonicity. For two graphs G_1 and G_2, we write $G_1 \subseteq G_2$ if $\mathrm{dom}(G_1) \subseteq \mathrm{dom}(G_2)$ and $R^{G_1} \subseteq R^{G_2}$ for every R. A path query q is called monotone if $G_1 \subseteq G_2$ implies $q(G_1) \subseteq q(G_2)$.

Lemma III.5. *Let e be an expression in $\mathcal{N}(di, ^{-1})$.*

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

Proof. 1) Let G be a graph and let $n \geq 3$ such that $G \subseteq K_n$. By Lemma III.4, we have $e(K_n) = \emptyset$. Hence, because e is monotone, also $e(G) = \emptyset$.

2) We can write $e = \cup_{i=1}^{n} e_i$ as a union of union-free expressions, since union distributes over composition and converse. By Lemma III.4, there must exist i such that $e_i(K_3)$ is equal to $di(K_3)$. Furthermore, for any $j = 1, \ldots, n$, $e_i(K_3)$ cannot equal $id(K_3)$ or $all(K_3)$. If $e_j(K_3) = \emptyset$, then $e_j \equiv \emptyset$ by the previous case.
So we may assume that $e_j(K_3) = di(K_3)$ for $j = 1, \ldots, n$. Take such an e_j. We know $e_j \not\equiv id$ so e_j can be written as $H_1 \circ \ldots \circ H_l$ with $H_k \in \{R, R^{-1}, di\}$. Indeed, this is possible since $(R \circ S)^{-1} \equiv (S^{-1} \circ R^{-1})$. If $l \geq 2$, the first composition already yields all on K_3. Indeed, $R^{-1}(K_3) = R(K_3) = all(K_3)$ and $R^2(K_3) = di \circ R(K_3) = R \circ di(K_3) = di^2(K_3) = all(K_3)$. Composing $all(K_3)$ with $all(K_3)$ or $di(K_3)$ is again $all(K_3)$. Thus $e_j(K_3) = all(K_3)$ which is impossible. Hence, $l = 1$. Here, H_1 has to be di, because $R(K_3) = R^{-1}(K_3) = all(K_3)$.

3) Similar to the previous case.

□

Let us now look at the outcome of expressions on the graph Z.

Lemma III.6. *Let $e \not\equiv \emptyset$ be a union-free expression in $\mathcal{N}(di, ^{-1})$.*

1) *If di occurs in e, then $di(Z) \cap e(Z) \neq \emptyset$.*
2) *If di does not occur in e, or it occurs at least twice, then $id(Z) \cap e(Z) \neq \emptyset$.*

Proof. 1) Write $e = q_1 \circ di \circ q_2$ where q_1 is di-free. Note that q_1 or q_2 may be id. Since di-free expressions can be evaluated in a loop, $(1,1)$ is in $q_1(Z)$, whence $(1,2) \in q_1 \circ di(Z)$. Furthermore, if there is an odd number of di occurrences in q_2, $(2,3) \in q_3(Z)$, and otherwise $(2,2)$ in $q_2(Z)$. Indeed, every di-free sub expression can be evaluated in a loop, and on every di application, one can jump from 2 to 3 and vice versa. We may thus conclude that $(1,2)$ or $(1,3)$ is in $e(Z)$.

2) If e contains no di applications, then $e = R^n$ on Z for some positive n, since e is not equivalent to $\emptyset$ and Z is symmetrical. Hence, $e(Z) = R(Z) = id(Z)$.
If e contains at least two di applications, then we can write $e = q_1 \circ di \circ q_2 \circ di \circ q_3$ so that q_1 and q_3 are di-free. Now $(1,1)$ is in $q_1(Z)$ and in $q_3(Z)$. Hence $(1,2) \in q_1 \circ di(Z)$ and $(3,1)$ and $(2,1)$ in $di \circ q_3(Z)$. When di occurs an odd number of times in q_2, then $(2,3) \in q_3(Z)$; when it occurs an even number of times, $(2,2) \in q_2(Z)$. We may thus conclude that $(1,1) \in e(Z)$.

□

We next look at the outcome of expressions on the graph H.

Lemma III.7. *Let $e \not\equiv \emptyset$ be a union-free expression in $\mathcal{N}(di, ^{-1})$.*

1) *If $e \not\equiv di$ and di occurs exactly once in e, then $id(H) \cap e(H) \neq \emptyset$.*
2) *If $e \not\equiv id$ and e is di-free, then $di(H) \cap e(H) \neq \emptyset$.*

Proof. 1) Write $e = e_1 \circ di \circ e_2$ where e_1 and e_2 are di-free. One of e_1 or e_2 may be id, but not both, since $e \not\equiv di$. We will now consider all the possible scenarios for e_1 and e_2. Note that on H, every nonempty di-free expression q can be evaluated in a loop, i.e., $(1,1), (3,3) \in q(H)$.

- If $e_1 = q_1 \circ R$ where q_1 is di-free, then $(1,1) \circ (1,2) \circ (2,1) \circ (1,1) \in q_1 \circ R \circ di \circ e_2(H)$. Hence $(1,1) \in e_1 \circ di \circ e_2(H)$.
- If $e_2 = R^{-1} \circ q_2$ where q_2 is di-free, then $(1,1) \circ (1,2) \circ (2,1) \circ (1,1) \in e_1 \circ di \circ R^{-1} \circ q_2(H)$. Hence $(1,1) \in e_1 \circ di \circ e_2(H)$.
- If $e_1 = R^{-1} \circ R^{-n}$ and $e_2 = id$ where n may be zero, then $(2,1) \circ (1,1) \circ (1,2) \in R^{-1} \circ R^{-n} \circ di(H)$. Hence $(2,2) \in e_1 \circ di \circ e_2(H)$.
- If $e_1 = id$ and $e_2 = R^n \circ R$ where n may be zero, then $(2,1) \circ (1,1) \circ (1,2) \in di \circ R^n \circ R(H)$. Hence $(2,2) \in e_1 \circ di \circ e_2(H)$.
- If $e_1 = R^{-1} \circ R^{-n}$ and $e_2 = R^m \circ R$ where n and m may be zero, then $(2,1) \circ (1,1) \circ (1,3) \circ (3,3) \circ (3,2) \in R^{-1} \circ R^{-n} \circ di \circ R^m \circ R(H)$. Hence $(2,2) \in e_1 \circ di \circ e_2(H)$.
- If $e_1 = q_1 \circ R \circ R^{-1} \circ R^{-n}$ where n may be zero, then $(1,1) \circ (1,2) \circ (2,3) \circ (3,3) \circ (3,1) \in q_1 \circ R \circ R^{-1} \circ R^{-n} \circ di(H)$. Hence $(1,1) \in e_1 \circ di \circ e_2(H)$.
- If $e_2 = R^n \circ R \circ R^{-1} \circ q_2$ where n may be zero, then $(3,1) \circ (1,1) \circ (1,2) \circ (2,3) \circ (3,3) \in di \circ R^n \circ R \circ R^{-1} \circ q_2(H)$. Hence $(3,3) \in e_1 \circ di \circ e_2(H)$.

2) There are three possibilities.
- If e can be written as $q_1 \circ R \circ R^{-1} \circ q_2$, where q_1 and q_2 may be id, then $(1,1) \circ (1,2) \circ (2,3) \circ (3,3) \in q_1 \circ R \circ R^{-1} \circ q_2(H)$. Hence, $(1,3) \in e(H)$.
- If $e = R^n \circ R$ where n may be zero, then $(1,1) \circ (1,2) \in R^n \circ R(H)$. Hence $(1,2) \in e(H)$.
- If e can be written as $R^{-1} \circ q$ where q may be id, then $(2,1) \circ (1,1) \in R^{-1} \circ q(H)$. Hence $(2,1) \in e(H)$.

□

We are now ready for the

Proof of Proposition III.3. Let us denote the boolean query $R \circ \pi_1(R) \subseteq id$ by Q. Suppose for the sake of contradiction that the conjunction $e_1 \subseteq f_1 \wedge \cdots \wedge e_n \subseteq f_n$ expresses Q. We assume that no containment is trivial (a *trivial* containment is always true). Notice that $Q(K_3) = false$. Thus there exists $1 \leq i \leq n$ such that $e_i(K_3) \not\subseteq f_i(K_3)$. Hence $f_i(K_3) \neq all(K_3)$. In the remainder of the proof we will only work on the graphs K_3, H and Z, whence we can replace $^+$ with unions of compositions. We know that $f_i(K_3)$ is either $\emptyset$, $id(K_3)$, or $f_i(K_3) = di(K_3)$. We will now cover each of these scenarios and obtain a contradiction.

If $f_i(K_3) = \emptyset$, then $f_i \equiv \emptyset$ by Lemma III.5. Since $Q(Z) = true$, it must be that $e_i(Z) \subseteq f_i(Z)$. Thus $e_i(Z) = \emptyset$, whence $e_i \equiv \emptyset$ by Lemma III.6. This, however, contradicts that $e_i \subseteq f_i$ is not trivial.

If $f_i(K_3) = di(K_3)$, then $f_i \equiv di$ by Lemma III.5. Write $e_i = \cup_{j=1}^m g_j$ with g_j union-free. Since $e_i \subseteq f_i$ is not trivial, there has to exist $1 \leq j \leq m$ such that $g_j \not\equiv di$. If $g_j \equiv id$, then certainly $e_i(Z) \not\subseteq di(Z) = f_i(Z)$. The only case left to consider is that $g_j \not\equiv di$ and $g_j \not\equiv id$. If g_j contains zero or more than two di applications, then $e_i(Z) \cap id(Z) \neq \emptyset$ by Lemma III.6, whence $e_i(Z) \not\subseteq di(Z) = f_i(Z)$. This, however, contradicts that $Q(Z) = true$. On the other hand, if g_j contains exactly one di application, then $e_i(H) \cap id(H) \neq \emptyset$ by Lemma III.7, whence $e_i(H) \not\subseteq di(H) = f_i(H)$. This, however, contradicts that $Q(H) = true$.

If $f_i(K_3) = id(K_3)$, then $f_i \equiv id$ by Lemma III.5. Again write $e_i = \cup_{j=1}^m g_j$ with g_j union-free. Since $e_i \subseteq f_i$ is not trivial, there has to exist $1 \leq j \leq m$ such that $g_j \not\equiv id$. If g_j contains at least one di application, then $g_j(Z) \cap di(Z) \neq \emptyset$ by Lemma III.6, whence $e_i(Z) \not\subseteq id(Z) = f_i(Z)$. However, this contradicts that $Q(Z) = true$. On the other hand, if g_j is di-free, then $g_j(H) \cap di(H) \neq \emptyset$ by Lemma III.7, whence $e_i(H) \not\subseteq id(H) = f_i(H)$. However, this contradicts that $Q(H) = true$. □

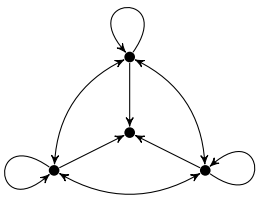

Fig. 5. Graph used in the proof of Proposition IV.1.

IV. Coprojection

Up to completion, there are two maximal fragments lacking coprojection: $\{di, ^{-1}, \cap, ^+\}$ and $\{-, ^+\}$. For the latter fragment, $\{\overline{\pi}\}^{\wedge\subseteq} \not\subseteq \{-, ^+\}^{\wedge\subseteq}$ follows directly from Proposition III.1, since $\pi \in \{\overline{\pi}\}$. For the former fragment, we show:

Proposition IV.1. *Let R be a relation name. The boolean query $\pi_1(R) \subseteq \pi_1(R \circ \overline{\pi}_1(R))$ is not in $\{di, ^{-1}, \cap, ^+\}^{\wedge\subseteq}$.*

Proof. Let us denote the boolean query $\pi_1(R) \subseteq \pi_1(R \circ \overline{\pi}_1(R))$ by Q. Let G be the graph in Figure 5. Observe that Q is true on G but false on K_3.

Suppose for the sake of contradiction that the conjunction $e_1 \subseteq f_1 \wedge \cdots \wedge e_n \subseteq f_n$ expresses Q. Since $Q(K_3) = false$ there exists $1 \leq i \leq n$ such that $e_i(K_3) \not\subseteq f_i(K_3)$. Hence $f_i(K_3) \neq all(K_3)$. In the remainder of the proof we will only work on the graphs K_3, K_4 and G, whence we can replace $^+$ with unions of compositions.

Since path queries in $\mathcal{N}(di, ^{-1}, \cap)$ are monotone, we have $f_i(K_3) \subseteq f_i(G) \subseteq f_i(K_4)$. This will be used a number of times.

Since $f_i(K_3) \neq all(K_3)$, the only possibilities for $f_i(K_3)$ are $id(K_3)$, $di(K_3)$, and $\emptyset$.

If $f_i(K_3) = id(K_3)$, then also $f_i(K_4) = id(K_4)$ by Lemma III.4. Hence, $f_i(G) \cap di(G) = \emptyset$. Since $Q(K_3) = false$, we have $e_i(K_3) \not\subseteq f_i(K_3)$, so $e_i(K_3) \cap di(K_3) \neq \emptyset$ whence also $e_i(G) \cap di(G) \neq \emptyset$. Thus $e_i(G) \not\subseteq f_i(G)$ which contradicts that $Q(G) = true$.

If $f_i(K_3) = di(K_3)$, this case is analogous to the previous case.

Fig. 6. Graph used in the proof of Proposition V.1.

Finally, if $f_i(K_3) = \emptyset$, then also $f_i(K_4) = \emptyset$ by Lemma III.4, whence also $f_i(G) = \emptyset$. Since $Q(K_3) = \mathit{false}$, we have $e_i(K_3) \not\subseteq \emptyset$. Hence also $e_i(G) \not\subseteq \emptyset$ which contradicts that $Q(G) = \mathit{true}$. □

V. Intersection

Up to completion, the unique maximal fragment lacking intersection is $\{di, \overline{\pi}, {}^{-1}, {}^{+}\}$. We are going to show:

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

To prove this proposition it will suffice to reason on the finite graphs K_3 and Z from Figure 4, and the graph ℓ_2 shown in Figure 6. We begin by showing that on these three graphs, projection and coprojection can be eliminated.

Lemma V.2. *Let e be an expression in $\mathcal{N}(di, \overline{\pi}, {}^{-1})$. Then, $\pi_i(e)$, for $i = 1, 2$, is equivalent to $\emptyset$ or id on the three graphs K_3, ℓ_2 and Z simultaneously.*

Proof. In this proof, whenever we write "equivalent" we mean equivalent on the three graphs K_3, ℓ_2 and Z. We proceed by induction on e. In the base case, $\pi_i(\emptyset) = \emptyset$ and $\pi_i(R) = \pi_i(R^{-1}) = \pi_i(di) = id$ on all three graphs.

If $e = \overline{\pi}_j(e_1)$, then $\pi_i(\overline{\pi}_j(e_1)) \equiv id - \pi_j(e_1)$. By induction $\pi_j(e_1)$ is equivalent to id or $\emptyset$, whence $id - \pi_j(e_1)$ also.

If $e = e_1 \cup e_2$, then $\pi_i(e_1 \cup e_2) = \pi_i(e_1) \cup \pi_i(e_2)$. By induction $\pi_i(e_1)$ and $\pi_i(e_2)$ are equivalent to id or $\emptyset$. Clearly, $\pi_i(e_1) \cup \pi_i(e_2)$ is equivalent to $\emptyset$ when both $\pi_i(e_1)$ and $\pi_i(e_2)$ are equivalent to $\emptyset$. In all other cases, $\pi_i(e_1) \cup \pi_i(e_2)$ is equivalent to id.

If $e = e_1 \circ e_2$, there are two cases:

- $\pi_1(e_1 \circ e_2) = \pi_1(e_1 \circ \pi_1(e_2))$. By induction, $\pi_1(e_1 \circ \pi_1(e_2))$ equals $\pi_1(e_1 \circ id) = \pi_1(e_1)$ or $\pi_1(e_1 \circ \emptyset) = \emptyset$.
- $\pi_2(e_1 \circ e_2) = \pi_2(\pi_2(e_1) \circ e_2)$. By induction $\pi_2(\pi_2(e_1) \circ e_2)$ equals $\pi_2(id \circ e_2) = \pi_2(e_2)$ or $\pi_2(\emptyset \circ e_2) = \emptyset$.

□

Note that, since $\overline{\pi}(e) \equiv id - \pi(e)$, the above lemma also holds for $\overline{\pi}(e)$.

We next look at the outcome of expressions on the graph ℓ_2.

Lemma V.3. *Let e be a union-free expression in $\mathcal{N}(di, {}^{-1})$.*

1) *If e is di-free, $e \not\equiv id$ and $e \not\equiv \emptyset$, then $e(\ell_2) \cap di(\ell_2) \neq \emptyset$.*
2) *If di occurs exactly once in e and $e \not\equiv di$, then $e(\ell_2) \cap id(\ell_2) \neq \emptyset$.*

Proof. 1) Since ℓ_2 is symmetrical, the converse operator does nothing and we can write $e = R^k$, with k positive since $e \not\equiv id$. If k is odd, clearly $(1, 2) \in R^k(\ell_2)$. If k is even, $(1, 2) \in R^{k-1}(\ell_2)$ so $(1, 3) \in R^k(\ell_2)$.

2) First, we describe some outcome results for R^n on ℓ_2:
   - If n is odd, then $(1, 2)$, $(2, 1)$, $(2, 3)$ and $(3, 2)$ are in $R^n(\ell_2)$;
   - If n is even, then $(1, 1)$ and $(2, 2)$ are in $R^n(\ell_2)$;
   - If $n > 1$ is even, then $(1, 3)$ and $(3, 1)$ are in $R^n(\ell_2)$.

   Now write e as $R^n \circ di \circ R^m$, where n and m may be zero (but not both).
   - If n and m are both odd, then $(2, 1) \circ (1, 3) \circ (3, 2) \in R^n \circ di \circ R^m(\ell_2)$. Hence $(2, 2) \in e(\ell_2)$.
   - If n is even and m is odd, then $(1, 1) \circ (1, 2) \circ (2, 1) \in R^n \circ di \circ R^m(\ell_2)$, whence also $(1, 1) \in e(\ell_2)$.
   - If n is odd and m is even, this case is symmetrical to the previous case.
   - If n is even and m is even, then n or m is strictly greater than one. If $n > 1$, then $(1, 3) \circ (3, 1) \circ (1, 1) \in R^n \circ di \circ R^m(\ell_2)$, whence $(1, 1) \in e(\ell_2)$. The case $m > 1$ is symmetrical. □

We can now give the

Proof of Proposition V.1. Let us denote the boolean query $R^2 \cap R \subseteq id$ by Q. Observe that Q is false on K_3 but true on ℓ_2 and Z.

Suppose for the sake of contradiction that the conjunction $e_1 \subseteq f_1 \wedge \cdots \wedge e_n \subseteq f_n$ expresses Q. We assume no containment is trivial.

Since $Q(K_3) = \mathit{false}$, there exists $1 \leq i \leq n$ such that $e_i(K_3) \not\subseteq f_i(K_3)$. In particular, $f_i(K_3) \neq \mathit{all}(K_3)$. In the remainder of the proof we will only work on the graphs K_3, Z and ℓ_2, whence we can replace $^+$ with unions of compositions. Furthermore, by Lemma V.2, we can eliminate $\overline{\pi}$ and π. So we may assume that e_i and f_i are in $\mathcal{N}(di, {}^{-1})$. Since $f_i(K_3) \neq \mathit{all}(K_3)$ the three possibilities for $f_i(K_3)$ are $\emptyset$, $id(K_3)$ or $di(K_3)$. We will now cover these three possibilities and obtain a contradiction.

If $f_i(K_3) = di(K_3)$, then $f_i \equiv di$ by Lemma III.5. Write $e_i = \cup_{j=1}^m g_j$ with g_j union-free. Since $e_i \subseteq f_i$ is not trivial, there has to exists $1 \leq j \leq m$ such that $g_j \not\equiv di$. If g_j contains exactly one di application, then $e_i(\ell_2) \cap id(\ell_2) \neq \emptyset$ by Lemma V.3, whence $e_i(\ell_2) \not\subseteq di(\ell_2) = f_i(\ell_2)$. This, however, contradicts that $Q(\ell_2) = \mathit{true}$. On the other hand, if g_k is di-free or has more than one di application, then $e_i(Z) \cap id(Z) \neq \emptyset$ by Lemma III.6, whence $e_i(Z) \not\subseteq di(Z) = f_i(Z)$. This, however, contradicts that $Q(Z) = \mathit{true}$.

If $f_i(K_3) = id(K_3)$, then $f_i \equiv id$ by Lemma III.5. Again write $e_i = \cup_{j=1}^m g_j$ with g_j union-free. Since $e_i \subseteq f_i$ is not trivial, there has to exist $1 \leq j \leq m$ such that $g_j \not\equiv id$ and $g_j \not\equiv \emptyset$. If g_j is di-free, then $g_j(\ell_2) \cap di(\ell_2) \neq \emptyset$ by Lemma V.3, whence $e_i(\ell_2) \not\subseteq id(\ell_2) = f_i(\ell_2)$. This, however, contradicts that $Q(\ell_2) = \mathit{true}$. On the other hand, if g_j contains at least one di application, then $g_j(Z) \cap di(Z) \neq \emptyset$ by Lemma III.6, whence $e_i(Z) \not\subseteq id(Z) = f_i(Z)$. This, however, contradicts that $Q(Z) = \mathit{true}$.

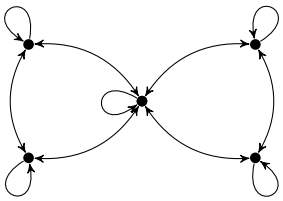

Fig. 7. Graph used in the proof of Proposition VI.1.

Finally, if $f_i(K_3) = \emptyset$, then $f_i \equiv \emptyset$. Since $Q(Z) = true$, we have $e_i(Z) \subseteq f_i(Z) = \emptyset$. Thus $e_i(Z) = \emptyset$, whence $e_i \equiv \emptyset$ by Lemma III.6 (we can again write e_i as a union of union-free expressions). This, however, contradicts that $e_i \subseteq f_i$ is not trivial. □

VI. Difference

Up to completion, the unique maximal fragment lacking difference is $\{\cap, \overline{\pi}, di, ^{-1}, ^{+}\}$, which we denote by NoDiff. We are going to show:

Proposition VI.1. *Let R be a relation name. The boolean query $id \subseteq R^2 \circ (R^2 - R) \circ R^2$ is not in* $\mathrm{NoDiff}^{\wedge\subseteq}$.

To prove this proposition it will suffice to reason on the complete graph K_3 and the bowtie graph B shown in Figure 7.

Lemma VI.2. *Every expression in $\mathcal{N}(\mathrm{NoDiff})$ is equivalent to $\emptyset$, id, di, R, $R \cap di$ or all on K_3 and B simultaneously.*

Proof. In this proof all equivalences are meant to hold on K_3 and B only. We proceed by structural induction on the expression e. For $e \in \{\emptyset, id, di, R\}$ the result is trivial. Note that we do not have to consider transitive closure, since on a fixed finite number of graphs, one can replace the transitive closure operator by a finite union of compositions.

Suppose $e = e_1 \cup e_2$. The only nontrivial cases are $e_1 = id$ and $e_2 = R \cap di$; $e_1 = id$ and $e_2 = R$; and $e_1 = di$ and $e_2 = R$. In the first case, $id \cup (R \cap di)(K_3) = R(K_3)$ and $id \cup (R \cap di)(B) = R(B)$. In the second case, $id \cup R(K_3) = R(K_3)$ and $id \cup R(B) = R(B)$. In the third case, $di \cup R(K_3) = all(K_3)$ and $di \cup R(B) = all(B)$.

Suppose $e = \overline{\pi}_i(e_1)$. If $e_1 \equiv \emptyset$ then $\overline{\pi}_i(e_1)(K_3) = id(K_3)$ and $\overline{\pi}_i(e_1)(B) = id(B)$. In any other case $\overline{\pi}_i(e_1)(K_3) = \emptyset$ and $\overline{\pi}_i(e_1)(B) = \emptyset$, since for any $g \in \{id, di, R, R \cap di, all\}$, we have $\overline{\pi}_i(g)(K_3) = \overline{\pi}_i(g)(B) = \emptyset$.

Suppose $e = e_1 \cap e_2$. Then the only nontrivial case is where $e_1 \equiv R$ and $e_2 \equiv id$. Here, $R \cap id(K_3) = id(K_3)$ and $R \cap id(B) = id(B)$ since K_3 and B both contain all self-loops.

Suppose $e = e_1 \circ e_2$. Since composing with $\emptyset$ results in $\emptyset$, and composing with id does nothing, we may focus on $e_1, e_2 \in \{di, R, R \cap di, all\}$. It is clear that $R \cap di(K_3) \subseteq e_i(K_3)$ and $R \cap di(B) \subseteq e_i(B)$. Hence $(R \cap di) \circ (R \cap di)(K_3) \subseteq e_1 \circ e_2(K_3)$ and $(R \cap di) \circ (R \cap di)(B) \subseteq e_1 \circ e_2(B)$. Therefore, since $(R \cap di) \circ (R \cap di)(K_3) = all(K_3)$ and $(R \cap di) \circ (R \cap di)(B) = all(B)$, we obtain $e_1 \circ e_2(K_3) = all(K_3)$ and $e_1 \circ e_2(B) = all(B)$.

The case $e = e_1^{-1}$ is trivial since all of the possible intermediate results are symmetrical. □

Proof of Proposition VI.1. Denote the boolean query $id \subseteq R^2 \circ (R^2 - R) \circ R^2$ by Q. Observe that Q is false on K_3 but true on B.

It suffices to show that a single containment $e_1 \subseteq e_2$ is never *false* on K_3 and *true* on B simultaneously. Indeed, this behavior is then preserved under conjunction.

By Lemma VI.2 e_1 and e_2 are equivalent to $\emptyset$, id, di, R, $R \cap di$ or all on K_3 and B simultaneously. From now on, equivalences are understood to be on K_3 and B only. We may assume that $e_1 \not\equiv \emptyset$ and $e_2 \not\equiv all$, since otherwise, the query expressed by $e_1 \subseteq e_2$ is the trivial true query.

If e_2 is $\emptyset$, id or di, then by Lemma VI.2 we have $e_1(K_3) \subseteq e_2(K_3)$ iff $e_1(B) \subseteq e_2(B)$. Hence the query $e_1 \subseteq e_2$ cannot distinguish K_3 and B.

If $e_2 \equiv R$, then again by Lemma VI.2 we have $e_1(K_3) \subseteq R(K_3)$ iff $e_1(B) \subseteq R(B)$, except for the case where $e_1 \equiv di$ or $e_1 \equiv all$. However, in these cases, $e_1(K_3) \subseteq e_2(K_3)$.

If $e_2 \equiv R \cap di$, then again by Lemma VI.2 we clearly have $e_1(K_3) \subseteq R \cap di(K_3)$ iff $e_1(B) \subseteq R \cap di(B)$ except maybe for the case where $e_1 \equiv di$. However, in that case, again, $e_1(K_3) \subseteq e_2(K_3)$. □

VII. Transitive closure

It seems obvious that transitive closure must be primitive, as it is the only operator that is not first-order definable. However, we want to establish primitivity across all fragments and all vocabularies. Thereto we would ideally like to find a boolean query over a single relation name that is not first-order expressible, but is expressible as a containment statement $e \subseteq f$ with e and f in $\mathcal{N}(^{+})$. Obvious candidates, such as connectivity or cyclicity, seem not expressible in this manner, however. In other contexts, transitive closure may even *not* be primitive. For example, every boolean query over a single relation name that is expressible as the nonemptyness of an expression in $\mathcal{N}(di, \pi, ^{+})$ is already expressible without using transitive closure [16].

Nevertheless, we have found that the simple boolean query "every node lies on a cycle" satisfies our needs:

Proposition VII.1. *Let R be a relation name. The boolean query $id \subseteq R^{+}$ is not first-order expressible.*

It follows that transitive closure is primitive. Proving this proposition is an exercise in Hanf locality [21], which requires finding the right graphs. We found the graphs G_1^ℓ and G_2^ℓ shown in Figure 8. In G_1^ℓ, every node lies on a cycle, but not in G_2^ℓ. Yet, for every natural number k and every $\ell > k$, the graphs G_1^ℓ and G_2^ℓ have the same k-neighborhood types with the same multiplicities, as summarized in Figure 9. Since first-order logic is Hanf-local, this implies that the boolean query $id \subseteq R^{+}$ is not first-order expressible.

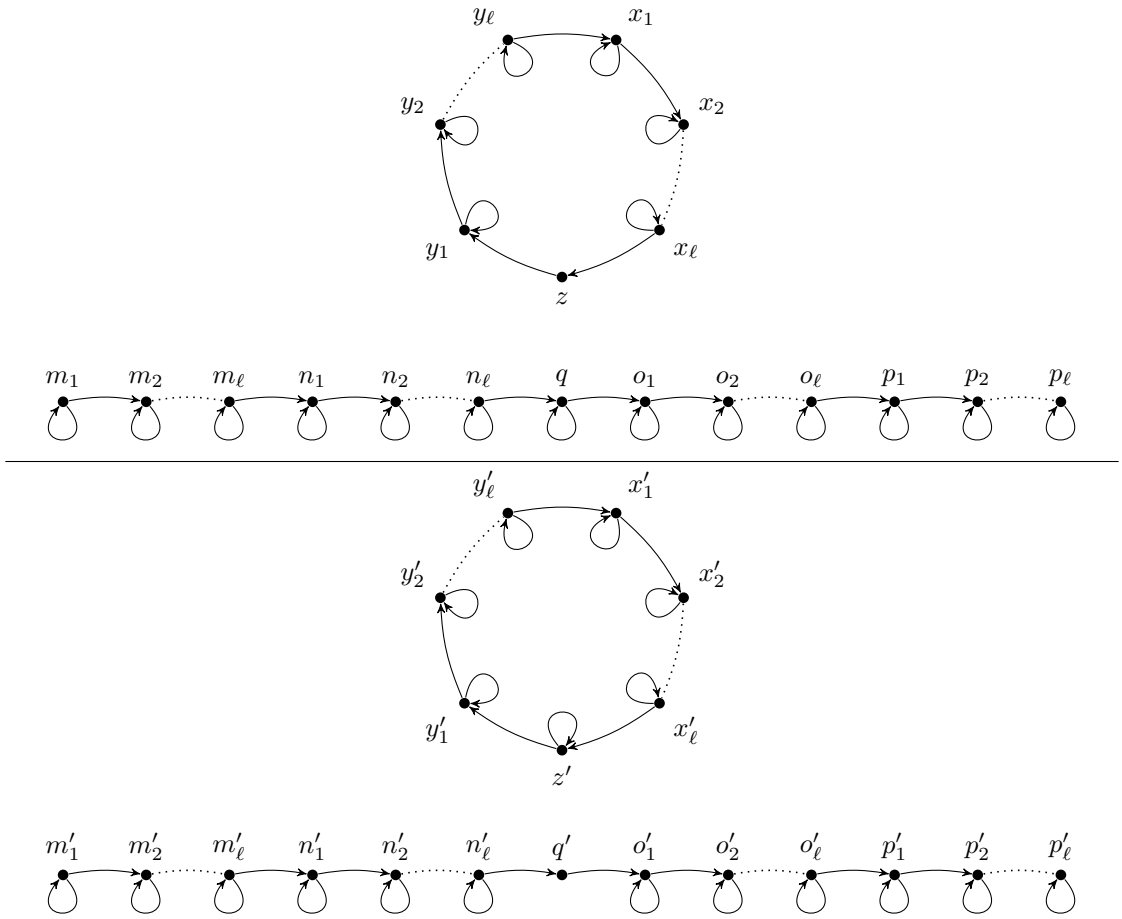

Fig. 8. Graphs G_1^ℓ (top) and G_2^ℓ (bottom) used in the proof of Proposition VII.1.

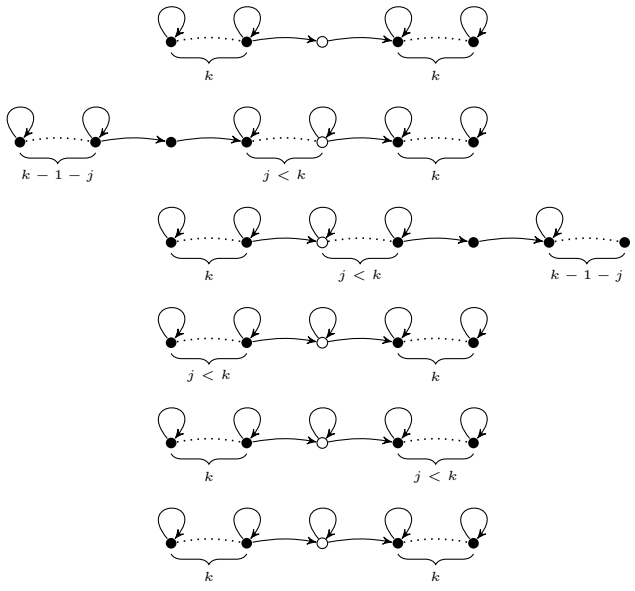

Fig. 9. k-neighborhood types. The white node indicates the center of the neighborhood. Except for the bottom type, each type occurs exactly once in G_1^ℓ and in G_2^ℓ with $\ell > k$ (and letting j range from 0 to $k-1$). The bottom type occurs exactly $6\ell - 4k + 1$ times in both graphs.

VIII. The full relation

Up to completion, the unique maximal fragment lacking *all* is $\{^{-1}, -, ^{+}\}$, which we denote by NoAll. We are going to show:

Proposition VIII.1. *Let R be a relation name. The boolean query $all \subseteq R$ is not in* $\mathrm{NoAll}^{\wedge\subseteq}$.

This proposition can be proven easily from the additivity of path queries expressible in $\mathcal{N}(\mathrm{NoAll})$. A path query q is called *additive* if for any two graphs G_1 and G_2 such that $\mathrm{dom}(G_1)$ and $\mathrm{dom}(G_2)$ are disjoint, $q(G_1 \cup G_2) = q(G_1) \cup q(G_2)$.

Lemma VIII.2. *Every path query expressible in $\mathcal{N}(\mathrm{NoAll})$ is additive.*

This lemma follows from the additivity of connected stratified Datalog [22]. A direct proof is also possible.

Proof of Proposition VIII.1. Denote the boolean query $all \subseteq R$ by Q. Let G_1 and G_2 be two disjoint graphs, each consisting of just a single self-loop. Observe that Q is true on G_1 and G_2 but false on $G_1 \cup G_2$.

Suppose for the sake of contradiction that the conjunction $e_1 \subseteq f_1 \wedge \cdots \wedge e_n \subseteq f_n$ expresses Q. Since $Q(G_1 \cup G_2) =$

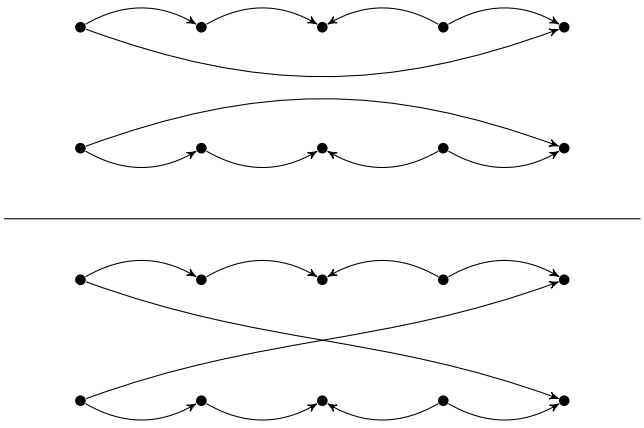

Fig. 10. Graphs used in the proof of Proposition X.1.

false, there exists $1 \leq j \leq n$ such that $e_j(G_1 \cup G_2) \not\subseteq f_j(G_1 \cup G_2)$. By additivity, $e_j(G_1) \cup e_j(G_2) \not\subseteq f_j(G_1) \cup f_j(G_2)$. Hence, $e_j(G_1) \not\subseteq f_j(G_1)$ or $e_j(G_2) \not\subseteq f_j(G_2)$, which contradicts that Q is true on both G_1 and G_2. □

IX. Diversity

Up to completion, there are two maximal fragments lacking diversity: $\{^{-1}, -, ^{+}\}$ and $\{^{-1}, all, \overline{\pi}, \cap, ^{+}\}$. We will show that in neither fragment, the boolean query $di \subseteq \emptyset$ ("there is only one node") is expressible as a conjunction of containments.

The fragment $\{^{-1}, -, ^{+}\}$ has set difference, so using Proposition III.2, we can invoke our previous work on nonemptiness queries. Indeed, it has already been shown [14, Proposition 5.4(1)] that the boolean query $di = \emptyset$ can not be expressed as the emptiness of an expression in $\mathcal{N}(^{-1}, -, ^{+})$.

For the other fragment, there is a simple direct proof.

Proposition IX.1. *The boolean query $di \subseteq \emptyset$ is not in $\{^{-1}, all, \overline{\pi}, \cap, ^{+}\}^{\wedge\subseteq}$.*

Proof. Let G be the graph consisting of a single self-loop. The boolean query is true on G but false on K_3. However, every expression in $\mathcal{N}(^{-1}, all, \overline{\pi}, \cap, ^{+})$ is equivalent to id, all or $\emptyset$ on G and K_3 simultaneously, which immediately implies the proposition.

The above claim is readily verified by induction. Indeed, the base case is trivial, and the induction step readily follows since the set $\{all, id, \emptyset\}$ is closed under all operators in the fragment. □

X. Converse

Up to completion, the unique maximal fragment lacking converse is $\{di, -, ^{+}\}$. We show:

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

To prove this proposition it will suffice to reason only on the two graphs G_1 (top) and G_2 (bottom) shown in Figure 10. We recall:

Lemma X.2 ([14, Proposition 6.6])**.** *$e(G_1) \neq \emptyset$ implies $e(G_2) \neq \emptyset$ for every expression e in $\mathcal{N}(di, -)$.*

With this lemma in hand we can give the

Proof of Proposition X.1. Let us denote the boolean query $R^2 \circ R^{-1} \circ R \subseteq R \cup R^2$ by Q. Observe that Q is true on G_1 but false on G_2. Suppose for the sake of contradiction that Q is in $\{di, -, ^{+}\}^{\wedge\subseteq}$. Then by Proposition III.2 Q is also expressible as $e = \emptyset$ with e in $\mathcal{N}(di, -, ^{+})$. Reasoning only on the two finite graphs G_1 and G_2, we may assume e does not use transitive closures, as we can replace these by unions of compositions. By assumption, $e(G_1)$ is empty but $e(G_2)$ is not. Equivalently, $e'(G_1) \neq \emptyset$ but $e'(G_2) = \emptyset$, with e' the expression $all - (all \circ e \circ all)$. This, however, contradicts Lemma X.2. □

XI. Conclusion

There are several directions for further work. There are two popular ways of expressing boolean queries: by nonemptiness statements, and by conjunctions of containment statements. Now that we understand the expressive power of these two approaches, we can also start comparing them. We have already largely completed this work, except for one technical open question, which asks whether every boolean query expressible by the nonemptiness of an expression $\mathcal{N}(\pi)$ belongs to $\{di, ^{-1}\}^{\wedge\subseteq}$.

The reader may have noticed that all our primitivity results have been proven using boolean queries expressible as a single containment. Indeed, a natural question is whether conjunctions actually add expressive power. For fragments with set difference the answer is of course negative by Proposition III.2. For the fragments $\{di, ^{-1}, ^{+}\}$ and $\{\cap, ^{-1}, ^{+}\}$ we can show that containment statements are not closed under conjunction. For the former fragment, the conjunction $R^3 \subseteq id \wedge R^2 \subseteq R$ is not expressible as a single containment; for the latter fragment, this holds for the conjunction $R^3 \subseteq \emptyset \wedge R^2 \subseteq R$. But this question remains largely open.

The attractiveness of using containment statements is that it allows to express nonmonotone boolean queries using a monotone language. Indeed, just stating $R \subseteq S$, for relation names R and S, is already a nonmonotone boolean query.[1] Vice versa, in comparing nonemptiness to containment statements, it would be a powerful tool to have a full understanding of monotone queries expressible as (conjunctions of) containment statements. A first result we could prove in this direction is that every monotone boolean query expressible as a containment of $\mathcal{N}(all)$ expressions is in fact expressible as the nonemptiness of an $\mathcal{N}(all)$ expression. This may be seen as a preservation theorem, similar to, say, the theorem that the monotone first-order boolean queries are those expressible by positive-existential sentences (allowing nonequalities) [23].

[1] As already recalled before Lemma III.5, for two graphs G_1 and G_2, we write $G_1 \subseteq G_2$ if $\mathrm{dom}(G_1) \subseteq \mathrm{dom}(G_2)$ and $R^{G_1} \subseteq R^{G_2}$ for every R. A boolean query q is called monotone if whenever $G_1 \subseteq G_2$ and q is true on G_1, it is also true on G_2.

Finally, a really useful and interesting operator which we have not covered in this paper is the residual [3]. It expresses a natural form of universal quantification and its expressive power relative to other operators is largely unexplored. We also do not know much about basic reasoning tasks, such as deciding satisfiability or subsumption, in the basic algebra extended with residual.

ACKNOWLEDGMENT

We thank the anonymous reviewers for careful reading and useful comments.

REFERENCES

[1] A. Tarski, "On the calculus of relations," *Journal of Symbolic Logic*, vol. 6, pp. 73–89, 1941.

[2] A. Tarski and S. Givant, *A Formalization of Set Theory Without Variables*, ser. AMS Colloquium Publications. American Mathematical Society, 1987, vol. 41.

[3] V. Pratt, "Origins of the calculus of binary relations," in *Proceedings 7th Annual IEEE Symposium on Logic in Computer Science*, 1992, pp. 248–254.

[4] R. Maddux, *Relation Algebras*. Elsevier, 2006.

[5] K. Ng, "Relation algebras with transitive closure," Ph.D. dissertation, University of California, Berkeley, 1984.

[6] R. Hirsch and I. Hodkinson, *Relation Algebras by Games*. Elsevier, 2002.

[7] M. Marx and M. de Rijke, "Semantic characterizations of navigational XPath," *SIGMOD Record*, vol. 34, no. 2, pp. 41–46, 2005.

[8] B. ten Cate and M. Marx, "Navigational XPath: Calculus and algebra," *SIGMOD Record*, vol. 36, no. 2, pp. 19–26, 2007.

[9] G. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, S. Vansummeren, and Y. Wu, "Relative expressive power of navigational querying on graphs," in *Proceedings 14th International Conference on Database Theory*, 2011.

[10] P. Wood, "Query languages for graph databases," *SIGMOD Record*, vol. 41, no. 1, pp. 50–60, Mar. 2012.

[11] L. Libkin, W. Martens, and D. Vrgoč, "Quering graph databases with XPath," in *Proceedings 16th International Conference on Database Theory*. ACM, 2013.

[12] R. Angles, P. Barceló, and G. Rios, "A practical query language for graph DBs," in *Proceedings 7th Alberto Mendelzon International Workshop on Foundations of Data Management*, ser. CEUR Workshop Proceedings, L. Bravo and M. Lenzerini, Eds., vol. 1087, 2013.

[13] R. Maddux, "The origin of relation algebras in the development and axiomatization of the calculus of relations," *Studia Logica*, vol. 50, no. 3/4, pp. 421–455, 1991.

[14] G. Fletcher, M. Gyssens, D. Leinders, D. Surinx, J. Van den Bussche, D. Van Gucht, S. Vansummeren, and Y. Wu, "Relative expressive power of navigational querying on graphs," *Information Sciences*, vol. 298, pp. 390–406, 2015.

[15] D. Surinx, G. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, S. Vansummeren, and Y. Wu, "Relative expressive power of navigational querying on graphs using transitive closure," *Logic Journal of the IGPL*, vol. 23, no. 5, pp. 759–788, 2015.

[16] G. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, S. Vansummeren, and Y. Wu, "The impact of transitive closure on the expressiveness of navigational query languages on unlabeled graphs," *Annals of Mathematics and Artificial Intelligence*, vol. 73, no. 1–2, pp. 167–203, 2015.

[17] G. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, and S. Vansummeren, "Similarity and bisimilarity notions appropriate for characterizing indistinguishability in fragments of the calculus of relations," *Journal of Logic and Computation*, vol. 25, no. 3, pp. 549–580, 2015.

[18] M. Marx and Y. Venema, *Multi-Dimensional Modal Logic*. Springer, 1997.

[19] D. Kozen, "Kleene algebra with tests," *ACM Transactions on Programming Languages and Systems*, vol. 19, no. 3, pp. 427–443, 1997.

[20] P. Brunet and D. Pous, "Petri automata for Kleene allegories," in *Proceedings 30th Annual ACM/IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 2015, pp. 68–79.

[21] L. Libkin, *Elements of Finite Model Theory*. Springer, 2004.

[22] T. Ameloot, B. Ketsman, F. Neven, and D. Zinn, "Weaker forms of monotonicity for declarative networking: A more fine-grained answer to the CALM-conjecture," *ACM Transactions on Database Systems*, vol. 40, no. 4, p. article 21, 2016.

[23] M. Benedikt, J. Leblay, B. ten Cate, and E. Tsamoura, *Generating plans from proofs: The interpolation-based approach to query reformulation*. Morgan&Claypool, 2016.