Information Sciences 298 (2015) 390–406

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Relative expressive power of navigational querying on graphs ☆

George H.L. Fletcher [a], Marc Gyssens [b], Dirk Leinders [b], Dimitri Surinx [b], Jan Van den Bussche [b,*], Dirk Van Gucht [c], Stijn Vansummeren [d], Yuqing Wu [c]

[a] Eindhoven University of Technology, The Netherlands
[b] Hasselt University & Transnational University of Limburg, Belgium
[c] Indiana University, United States
[d] Université Libre de Bruxelles, Belgium

ARTICLE INFO

Article history:
Received 2 January 2014
Received in revised form 6 November 2014
Accepted 22 November 2014
Available online 3 December 2014

Keywords:
Graph databases
Query languages
Expressive power

ABSTRACT

Motivated by both established and new applications, we study navigational query languages for graphs (binary relations). The simplest language has only the two operators union and composition, together with the identity relation. We make more powerful languages by adding any of the following operators: intersection; set difference; projection; coprojection; converse; and the diversity relation. All these operators map binary relations to binary relations. We compare the expressive power of all resulting languages. We do this not only for general path queries (queries where the result may be any binary relation) but also for boolean or yes/no queries (expressed by the nonemptiness of an expression). For both cases, we present the complete Hasse diagram of relative expressiveness. In particular the Hasse diagram for boolean queries contains some nontrivial separations and a few surprising collapses.

1. Introduction

Graph databases, and the design and analysis of query languages appropriate for graph data, have a rich history in database systems and theory research [3]. Originally investigated from the perspective of object-oriented databases, interest in graph databases research has been continually renewed, motivated by data on the Web [2,16] and new applications such as dataspaces [19], Linked Data [8], and RDF [32].

Typical of access to graph-structured data is its navigational nature. Indeed, in restriction to trees, there is a standard navigational query language, called XPath, whose expressive power has been intensively studied [7,25]. XPath has been formalized in terms of a number of basic operators on binary relations [26]. Hence a natural approach [30,22,1] is to take this same set of operators but now evaluate them over graphs instead of over trees. Our goal in this paper is to understand the relative importance of the different operators in this setting.

Concretely, in the present paper, we consider a number of natural operators on binary relations (graphs): union; composition; intersection; set difference; projection; coprojection; converse; and the identity and diversity relations. While some of these operators also appear in XPath, they are there evaluated on trees. The largest language that we consider has all operators, while the smallest language has only union, composition, and the identity relation. When a language has set difference,

☆ An extended abstract announcing the results of this paper was presented at the 14th International Conference on Database Theory, Uppsala, Sweden, March 2011.

* Corresponding author.

http://dx.doi.org/10.1016/j.ins.2014.11.031
0020-0255/

it also has intersection, by $R \cap S = R - (R - S)$. Interestingly, the ensemble of all operators except intersection and set difference precisely characterizes the first-order queries safe for bisimulation [36,26]. This logical grouping of operators is also present in our research, where we often have to treat the case without intersection separately from the case with intersection.[1]

Just as in the relational algebra, expressions are built up from input relation names using these operators. Since each operator maps binary relations to binary relations, these query languages express queries from binary relations to binary relations: we call such queries *path queries*. By identifying nonemptiness with the boolean value 'true' and emptiness with 'false', as is standard in database theory [4], we can also express yes/no queries within this framework. To distinguish them from general path queries, we shall refer to the latter as *boolean queries*.

The contribution of the present paper is providing a complete comparison of the expressiveness of all resulting languages, and this both for general path queries and boolean queries. While establishing the relative expressiveness for general path queries did not yield particularly surprising results, the task for the case of boolean queries proved much more challenging. For example, consider the converse operator $R^{-1} = \{(y,x)|(x,y) \in R\}$. On the one hand, adding converse to a language not yet containing this feature sometimes adds boolean query power. This is, e.g., the case for the language containing all other features. The proof, however, is nontrivial and involves a specialized application of invariance under bisimulation known from arrow logics. On the other hand, adding converse to a language containing projection but not containing intersection does not add any boolean query power. We thus obtain a result mirroring similar results known for XPath on trees [7,28,37], where, e.g., downward XPath is known to be as powerful as full XPath for queries evaluated at the root.

Let us briefly discuss some of the methods we use. In many cases where we separate a language $\mathcal{L}_1$ from a language $\mathcal{L}_2$, we can do this in a strong sense: we are able to give a single counterexample, consisting of a pair (A, B) of finite binary relations such that A and B are distinguishable by an expression from $\mathcal{L}_1$ but indistinguishable by any expression from $\mathcal{L}_2$. Notice that in general, separation is established by providing an infinite sequence of relation pairs such that some expression from $\mathcal{L}_1$ distinguishes all pairs but no expression of $\mathcal{L}_2$ distinguishes all pairs. Existence of a single counterexample pair is therefore nonobvious, and we do not really know whether there is a deeper reason why in our setting this strong form of separation can often be established. Strong separation is desirable as it immediately implies separation of $\mathcal{L}_1$ not only from $\mathcal{L}_2$ but also from the infinitary variant of $\mathcal{L}_2$ (which allows infinite unions, as in infinitary logic [10]). Note that indistinguishability of a pair of finite binary relations can in principle be checked by computer, as the number of possible binary relations on a finite domain is finite. Indeed, in many cases we have used this "brute-force approach" to verify indistinguishability. In some cases, however, this approach is not feasible within a reasonable time. Fortunately, by applying invariance under bisimulation for arrow logics [27], we can alternatively check a sufficient condition for indistinguishability in polynomial time. We have applied this alternative approach in our computer checks. Finally, the cases where we could not establish strong separation fall in the class of conjunctive queries [4]. We developed a method based on homomorphism techniques to establish ordinary separation for these cases.

The languages considered here are very natural and date all the way back to the "calculus of relations" created by Peirce and Schröder, and popularized and greatly developed by Tarski and his collaborators [33,34]. The full language actually has the same expressive power as 3-variable first-order logic (FO^3) under the active-domain semantics, for path queries as well as for boolean queries. Due to the naturalness of the languages, they appear in many other fields where binary relations are important, such as description logics, dynamic logics, arrow logics, and relation algebras [6,21,27,9,23,20]. Thus, our results also yield some new insight into these fields. The investigation of expressive power as in the present paper is very natural from a database theory perspective. In the above-mentioned fields, however, one is primarily interested in other questions, such as computational complexity of model checking, decidability of satisfiability, and axiomatizability of equivalence. The expressiveness issues investigated in this paper have not been investigated before.[2]

At this point we must repeat that also in the database field, graph query languages have been investigated intensively. There is, for example, the vast body of work on conjunctive regular path queries (CRPQs) [5]. As a matter of fact, CRPQs are subsumed in the calculus of relations, with the exception of the Kleene star (transitive closure) operator. Indeed, the results reported in this journal article have been extended to the setting where transitive closure is present, as originally announced in our conference paper [13]. This extension will be elaborated in a companion journal article [12]; additional results on the special case of a single relation name have been published in a third journal article [14].

This paper is further organized as follows. In Section 2, we define the class of languages studied in the paper. In Section 3, we describe the techniques we use to separate one language from another. In Section 4 we present our two main technical results in a self-contained manner: first, the added power of projection in expressing boolean queries, compared to the language without intersection and coprojection; second, the elimination of converse in languages with projection, but without intersection. Then we establish the complete Hasse diagram of relative expressiveness. We do so for path queries in Section 5, and for boolean queries in Section 6. Finally, we discuss future research directions in Section 7.

[1] Strictly speaking, van Benthem's discussion [36] does not include the converse operator nor the identity and diversity relations.

[2] Strictly speaking, one may argue that the "calculus of relations" refers to a set of equational axioms now known as the axioms for relation algebras (see the references above). However, the original and natural interpretation of the operations of the calculus of relations is clearly that of operations on binary relations [34,31]. In modern terminology this interpretation corresponds to 'representable' relation algebras. We stress that the present paper focuses on the expressive power of the various operations and not on axiomatizability, completeness of equations, or representability of abstract relation algebras.

2. Preliminaries

In this paper, we are interested in navigating over graphs whose edges are labeled by symbols from a finite, nonempty set of labels Λ. We can regard these edge labels as binary relation names and thus regard Λ as a relational database schema. For our purposes, then, a *graph* G is an instance of this database schema Λ. That is, assuming an infinite universe V of data elements called *nodes*, G assigns to every $R \in \Lambda$ a relation $G(R) \subseteq V \times V$. Each pair in $G(R)$ is called an *edge* with label R. In what follows, $G(R)$ may be infinite, unless explicitly stated otherwise. All inexpressibility results in this paper already hold in restriction to finite graphs, however.

The most basic language for navigating over graphs we consider is the algebra $\mathcal{N}$ whose expressions are built recursively from the edge labels, the primitive $\emptyset$, and the primitive *id*, using composition ($e_1 \circ e_2$) and union ($e_1 \cup e_2$). Semantically, each expression $e \in \mathcal{N}$ defines a path query. A *path query* is a function q taking any graph G as input and returning a binary relation $q(G) \subseteq \text{adom}(G) \times \text{adom}(G)$. Here, $\text{adom}(G)$ denotes the *active domain* of G, which is the set of all entries occurring in one of the relations of G. Formally,

$$\text{adom}(G) = \{m | \exists n, \exists R \in \Lambda : (m,n) \in G(R) \vee (n,m) \in G(R)\}.$$

In detail, the semantics of $\mathcal{N}$ is inductively defined as follows:

$$R(G) = G(R);$$
$$\emptyset(G) = \emptyset;$$
$$id(G) = \{(m,m) | m \in \text{adom}(G)\};$$
$$e_1 \circ e_2(G) = \{(m,n) | \exists p ((m,p) \in e_1(G) \ \& \ (p,n) \in e_2(G))\};$$
$$e_1 \cup e_2(G) = e_1(G) \cup e_2(G).$$

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

$$di(G) = \{(m,n) | m,n \in \text{adom}(G) \ \& \ m \neq n\};$$
$$e^{-1}(G) = \{(m,n) | (n,m) \in e(G)\};$$
$$e_1 \cap e_2(G) = e_1(G) \cap e_2(G);$$
$$e_1 - e_2(G) = e_1(G) - e_2(G);$$
$$\pi_1(e)(G) = \{(m,m) | m \in \text{adom}(G) \ \& \ \exists n (m,n) \in e(G)\};$$
$$\pi_2(e)(G) = \{(m,m) | m \in \text{adom}(G) \ \& \ \exists n (n,m) \in e(G)\};$$
$$\overline{\pi}_1(e)(G) = \{(m,m) | m \in \text{adom}(G) \ \& \ \neg\exists n (m,n) \in e(G)\};$$
$$\overline{\pi}_2(e)(G) = \{(m,m) | m \in \text{adom}(G) \ \& \ \neg\exists n (n,m) \in e(G)\}.$$

If F is a set of nonbasic features, we denote by $\mathcal{N}(F)$ the language obtained by adding all features in F to $\mathcal{N}$. For example, $\mathcal{N}(\cap)$ denotes the extension of $\mathcal{N}$ with intersection, and $\mathcal{N}(\cap, \pi)$ denotes the extension of $\mathcal{N}$ with intersection and both projections.[3] We will see below that extending the basic algebra with diversity, difference, and converse is sufficient to express all other nonbasic features. This full language $\mathcal{N}(-, di, ^{-1})$ is known as the *calculus of relations*.

We will actually compare language expressiveness at the level of both path queries and boolean queries. Path queries were defined above; a *boolean query* is a function from graphs to $\{\text{true}, \text{false}\}$.

Definition 2.1. A path query q is expressible in a language $\mathcal{N}(F)$ if there exists an expression $e \in \mathcal{N}(F)$ such that, for every graph G, we have $e(G) = q(G)$. Similarly, a boolean query q is expressible in $\mathcal{N}(F)$ if there exists an expression $e \in \mathcal{N}(F)$ such that, for every graph G, we have that $e(G)$ is nonempty if, and only if, $q(G)$ is true. In both cases, we say that q *is expressed by* e.

In what follows, we write $\mathcal{N}(F_1) \leqslant^{\text{path}} \mathcal{N}(F_2)$ if every path query expressible in $\mathcal{N}(F_1)$ is also expressible in $\mathcal{N}(F_2)$. Similarly, we write $\mathcal{N}(F_1) \leqslant^{\text{bool}} \mathcal{N}(F_2)$ if every boolean query expressible in $\mathcal{N}(F_1)$ is also expressible in $\mathcal{N}(F_2)$. Note that $\mathcal{N}(F_1) \leqslant^{\text{path}} \mathcal{N}(F_2)$ implies $\mathcal{N}(F_1) \leqslant^{\text{bool}} \mathcal{N}(F_2)$, but not necessarily the other way around. We write $\not\leqslant^{\text{path}}$ and $\not\leqslant^{\text{bool}}$ for the negation of $\leqslant^{\text{path}}$ and $\leqslant^{\text{bool}}$.

Remark 2.2. The attentive reader will note that every fragment $\mathcal{N}(F)$ actually depends on the label vocabulary Λ which is arbitrary but fixed. So to be fully precise we would need to use the notation $\mathcal{N}_\Lambda(F)$. For all the results in this paper, a comparison of fragments of the form $\mathcal{N}(F_1) \leqslant \mathcal{N}(F_2)$ (with $\leqslant$ being $\leqslant^{\text{path}}$ or $\leqslant^{\text{bool}}$) can be interpreted to mean that we have $\mathcal{N}_\Lambda(F_1) \leqslant \mathcal{N}_\Lambda(F_2)$ for every Λ. Moreover, whenever we have a negative result of the form $\mathcal{N}(F_1) \not\leqslant \mathcal{N}(F_2)$, this will actually already hold for the simplest Λ consisting of a single label.

[3] We do not consider extensions of $\mathcal{N}$ in which only one of the two projections, respectively one of the two coprojections, is present.

To illustrate, in the interpretation described above, the *id* relation may be considered redundant in any fragment that includes the projections. Indeed, we can express *id* as $\bigcup_{R\in\Lambda}(\pi_1(R)\cup\pi_2(R))$. This observation falls outside the scope of the present investigation, however, since we do not consider *id* as an optional feature; it belongs to all fragments considered in this paper.

Remark 2.3. The language XPath [38] also includes the path equality operator $.[e_1 = e_2]$ (in XPath called 'general comparison'), with the following semantics:

$$.[e_1 = e_2](G) = \{(m,m) | m \in \mathrm{adom}(G)\ \&\ \exists n\,(m,n) \in e_1(G)\cap e_2(G)\}.$$

This operator can be expressed in the fragment $\mathcal{N}(\pi,\cap)$ as $\pi_1(e_1\cap e_2)$, as well as in the fragment $\mathcal{N}(^{-1},\cap)$ as $(e_1\cap e_2^{-1})\cap id$. Actually the latter expression is not particular to this example, because it reflects the way in which projection is expressed using converse and intersection, as we will see in Section 5.

3. Tools to establish separation

Our results in Sections 5 and 6 will use the following tools to separate a language $\mathcal{N}(F_1)$ from a language $\mathcal{N}(F_2)$, i.e., to establish that $\mathcal{N}(F_1)\not\leqslant^{\mathrm{path}}\mathcal{N}(F_2)$, or $\mathcal{N}(F_1)\not\leqslant^{\mathrm{bool}}\mathcal{N}(F_2)$. It will also be useful to consider stronger variants of $\not\leqslant^{\mathrm{path}}$ and $\not\leqslant^{\mathrm{bool}}$.

Definition 3.1. The language $\mathcal{N}(F_1)$ is *strongly separable from* the language $\mathcal{N}(F_2)$ *at the level of path queries* if there exists a path query q expressible in $\mathcal{N}(F_1)$ and a finite graph G, such that, for every expression $e\in\mathcal{N}(F_2)$, we have $q(G)\neq e(G)$. We write $\mathcal{N}(F_1)\not\leqslant^{\mathrm{path}}_{\mathrm{strong}}\mathcal{N}(F_2)$ in this case. Similarly, $\mathcal{N}(F_1)$ is *strongly separable from* $\mathcal{N}(F_2)$ *at the level of boolean queries* if there exists a boolean query q expressible in $\mathcal{N}(F_1)$ and two finite graphs G_1 and G_2, with $q(G_1)$ true and $q(G_2)$ false, such that, for every expression $e\in\mathcal{N}(F_2)$, $e(G_1)$ and $e(G_2)$ are both empty, or both nonempty. We write $\mathcal{N}(F_1)\not\leqslant^{\mathrm{bool}}_{\mathrm{strong}}\mathcal{N}(F_2)$ in this case.

3.1. Path separation

Since $\mathcal{N}(F_1)\leqslant^{\mathrm{path}}\mathcal{N}(F_2)$ implies $\mathcal{N}(F_1)\leqslant^{\mathrm{bool}}\mathcal{N}(F_2)$, also $\mathcal{N}(F_1)\not\leqslant^{\mathrm{bool}}\mathcal{N}(F_2)$ implies $\mathcal{N}(F_1)\not\leqslant^{\mathrm{path}}\mathcal{N}(F_2)$ by contraposition. In most instances, we can therefore establish separation at the level of general path queries by establishing separation at the level of boolean queries. In the cases where $\mathcal{N}(F_1)\not\leqslant^{\mathrm{path}}\mathcal{N}(F_2)$ although $\mathcal{N}(F_1)\leqslant^{\mathrm{bool}}\mathcal{N}(F_2)$, we identify a finite graph G and an expression e_1 in $\mathcal{N}(F_1)$ and show that, for each expression e_2 in $\mathcal{N}(F_2)$, $e_1(G)\neq e_2(G)$. Notice that we actually establish strong path separation in those cases.

3.2. Boolean separation

To establish separation at the level of boolean queries, we use the following techniques.

3.2.1. Brute-force approach

Two graphs G_1 and G_2 are said to be *distinguishable* at the boolean level in a language $\mathcal{N}(F)$ if there exists a boolean query q expressible in $\mathcal{N}(F)$ such that exactly one of $q(G_1)$ and $q(G_2)$ is true, and the other is false. If such a query does not exists, G_1 and G_2 are said to be *indistinguishable* in $\mathcal{N}(F)$.

Using this terminology, two languages $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ are *strongly separable* if there exist two finite graphs G_1 and G_2 that are distinguishable in $\mathcal{N}(F_1)$, but indistinguishable in $\mathcal{N}(F_2)$.

For two finite graphs G_1 and G_2, (in) distinguishability in a language $\mathcal{N}(F)$ can easily be machine-checked through the Brute-Force Algorithm described below.

First observe that $\mathrm{adom}(G_1)$ and $\mathrm{adom}(G_2)$ are finite since G_1 and G_2 are finite. Moreover, for any e in $\mathcal{N}(F)$, $e(G_1)\subseteq\mathrm{adom}(G_1)\times\mathrm{adom}(G_1)$ and $e(G_2)\subseteq\mathrm{adom}(G_2)\times\mathrm{adom}(G_2)$. Hence, $e(G_1)$ and $e(G_2)$ are finite and the set $\{(e(G_1),e(G_2))|e\in\mathcal{N}(F)\}$ is also finite. Clearly, G_1 is indistinguishable from G_2 if this set contains only pairs that are both empty or both nonempty.

The Brute-Force Algorithm computes the above set by first initializing the set

$$B = \{(id(G_1),id(G_2))\}\cup\{(di(G_1),di(G_2))\}\cup\{(G_1(R),G_2(R))|R\in\Lambda\}$$

(where $\{(di(G_1),di(G_2))\}$ is omitted if $di\notin F$). It then adds new pairs (R_1,R_2) to B by closing B pair-wise under the features in $\mathcal{N}(F)$. That is, for every binary operator $\otimes$ in $\mathcal{N}(F)$ and all pairs $(R_1,R_2),(S_1,S_2)$ in B the algorithm adds $(R_1\otimes S_1,R_2\otimes S_2)$ to B, and similarly for the unary operators. Since there are only a finite number of pairs, the algorithm is guaranteed to end. Of course, the worst-case complexity of this brute-force algorithm is exponential. Nevertheless, we have successfully checked indistinguishability using this Brute-Force Algorithm in many of the cases that follow.

3.2.2. Bisimulation

We will not always be able to use the methodology above to separate two languages. In particular, to establish that $\mathcal{N}(^{-1},\cap) \not\leqslant^{\text{bool}} \mathcal{N}(-,di)$ we will employ invariance results under the notion of bisimulation below. In essence, this notion is based on the notion of bisimulation known from arrow logics [27]. Below, we adapt this notion to the current setting.

We require the following preliminary definitions. Let $\mathbf{G} = (G, a, b)$ denote a *marked graph*, i.e., a graph G with $a, b \in \text{adom}(G)$. The *degree* of an expression e is the maximum depth of nested applications of composition, projection and coprojection in e. For example, the degree of $R \circ R$ is 1, while the degree of both $R \circ (R \circ R)$ and $\pi_1(R \circ R)$ is 2. Intuitively, the depth of e corresponds to the quantifier rank of the standard translation of e into FO^3. For a set of features F, $\mathcal{N}(F)_k$ denotes the set of expressions in $\mathcal{N}(F)$ of degree at most k.

In what follows, we are only concerned with bisimulation results regarding $\mathcal{N}(-,di)$. The following is an appropriate notion of bisimulation for this language.

Definition 3.2 (*Bisimilarity*). Let k be a natural number, and let $\mathbf{G}_1 = (G_1, a_1, b_1)$ and $\mathbf{G}_2 = (G_2, a_2, b_2)$ be marked graphs. We say that $\mathbf{G}_1$ is bisimilar to $\mathbf{G}_2$ up to depth k, denoted $\mathbf{G}_1 \simeq_k \mathbf{G}_2$, if the following conditions are satisfied:

Atoms $a_1 = b_1$ if and only if $a_2 = b_2$; and $(a_1, b_1) \in G_1(R)$ if and only if $(a_2, b_2) \in G_2(R)$, for every $R \in \Lambda$;
Forth if $k > 0$, then, for every c_1 in $\text{adom}(G_1)$, there exists some c_2 in $\text{adom}(G_2)$ such that both $(G_1, a_1, c_1) \simeq_{k-1} (G_2, a_2, c_2)$ and $(G_1, c_1, b_1) \simeq_{k-1} (G_2, c_2, b_2)$;
Back if $k > 0$, then, for every c_2 in $\text{adom}(G_2)$, there exists some c_1 in $\text{adom}(G_1)$ such that both $(G_1, a_1, c_1) \simeq_{k-1} (G_2, a_2, c_2)$ and $(G_1, c_1, b_1) \simeq_{k-1} (G_2, c_2, b_2)$.

Expressions in $\mathcal{N}(-,di)$ of depth at most k are invariant under bisimulation:

Proposition 3.3. *Let k be a natural number; let e be an expression in $\mathcal{N}(-,di)_k$; and let $\mathbf{G}_1 = (G_1, a_1, b_1)$ and $\mathbf{G}_2 = (G_2, a_2, b_2)$ be marked graphs. If $\mathbf{G}_1 \simeq_k \mathbf{G}_2$ then $(a_1, b_1) \in e(G_1) \iff (a_2, b_2) \in e(G_2)$.*

In other words, if $\mathbf{G}_1 \simeq_k \mathbf{G}_2$, then any expression of degree at most k either both selects (a_1, b_1) in G_1 and (a_2, b_2) in G_2, or neither of them. As such, the marked graphs $\mathbf{G}_1$ and $\mathbf{G}_2$ are *indistinguishable* by expressions in $\mathcal{N}(-,di)_k$. The proof of Proposition 3.3 is by a straightforward induction on e.

The following proposition states how we can use Proposition 3.3 to show that some boolean query is not expressible in $\mathcal{N}(-,di)_k$.

Proposition 3.4. *Let k be a natural number. A boolean query q is not expressible in $\mathcal{N}(-,di)_k$ if there exist graphs G_1 and G_2 such that $q(G_1)$ is true and $q(G_2)$ is false, and, for each pair $(a_1, b_1) \in \text{adom}(G_1)^2$, there exists $(a_2, b_2) \in \text{adom}(G_2)^2$ such that $(G_1, a_1, b_1) \simeq_k (G_2, a_2, b_2)$.*

We omit the straightforward proof; we note that the converse implication holds as well [15].

3.2.3. Homomorphism approach

To show that $\mathcal{N}(\pi) \not\leqslant^{\text{bool}} \mathcal{N}(^{-1}, di)$, we used an entirely different technique, based on the theory of conjunctive queries and the nonexistence of certain homomorphisms on particular graphs. The details are given in Section 4.1.

4. The power of various operators

In this section, two main technical results are shown regarding the power of various operators. The first result (Proposition 4.1) states that the π operator (in combination with the basic operators) provides some boolean querying power that cannot be provided by the $^{-1}$ and di operators. This is a sharp expressivity result on projection, since adding any other feature to the fragment $\mathcal{N}(^{-1}, di)$ leads to the expressibility of projection.

Proposition 4.1. $\mathcal{N}(\pi) \not\leqslant^{\text{bool}} \mathcal{N}(^{-1}, di)$.

Since this result is highly technical, it is proven in Section 4.1.

The second result (Proposition 4.2) shows that, at the level of boolean queries, $^{-1}$ does not add expressive power in the presence of π and in the absence of $\cap$.

Proposition 4.2. *Let F be a set of nonbasic features for which $- \notin F$ and $\cap \notin F$. Then, $\mathcal{N}(F \cup \{^{-1}\}) \leqslant^{\text{bool}} \mathcal{N}(F \cup \{\pi\})$.*

Example 4.3. To illustrate Proposition 4.2, consider the expression $e_1 = R^3 \circ R^{-1} \circ R^3$ in $\mathcal{N}(^{-1})$. The expression $\pi_1(e_1)$ can be equivalently expressed in $\mathcal{N}(\pi)$ as $\pi_1\left(R^3 \circ \pi_2(\pi_1(R^3) \circ R)\right)$. Now observe that, for any graph G, we have that $e_1(G)$ is nonempty if and only if $\pi_1(e_1)(G)$ is nonempty.

Using this same observation, one can express the non-emptiness of the expression $e_2 = R \circ \overline{\pi}_2((R \circ S) \cup (R^{-1} \circ S))$ in $\mathcal{N}(^{-1}, \overline{\pi})$ by the non-emptiness of the expression $\pi_1(e_2) = \pi_1(R \circ \overline{\pi}_2(R \circ S) \circ \overline{\pi}_2(\pi_1(R) \circ S))$ in $\mathcal{N}(\overline{\pi})$.

Proof of Proposition 4.2. Let e be an expression in $\mathcal{N}(F \cup \{^{-1}, \pi\})$. Without loss of generality, we may assume that $^{-1}$ is only applied in e to edge labels, so for each edge label R we also consider R^{-1} as an edge label. By simultaneous induction on the size of e (the number of nodes in the syntax tree), we prove for $i = 1, 2$ that

- $\pi_i(e)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$; and
- if $\overline{\pi} \in \overline{F}$, then $\overline{\pi}_i(e)$ is expressible in $\mathcal{N}(F)$.

Notice that the second statement is implied by the first, but we need to consider both statements together to make the induction work. The basis of the induction is trivial. For all operators except composition we reason as follows:

$$\pi_1(R^{-1}) = \pi_2(R) \qquad \overline{\pi}_1(R^{-1}) = \overline{\pi}_2(R)$$
$$\pi_2(R^{-1}) = \pi_1(R) \qquad \overline{\pi}_2(R^{-1}) = \overline{\pi}_1(R)$$
$$\pi_i(\pi_j(e')) = \pi_j(e') \qquad \overline{\pi}_i(\pi_j(e')) = \overline{\pi}_j(e')$$
$$\pi_i(\overline{\pi}_j(e')) = \overline{\pi}_j(e') \qquad \overline{\pi}_i(\overline{\pi}_j(e')) = \pi_j(e')$$
$$\pi_i(e_1 \cup e_2) = \pi_i(e_1) \cup \pi_i(e_2) \qquad \overline{\pi}_i(e_1 \cup e_2) = \overline{\pi}_i(e_1) \circ \overline{\pi}_i(e_2).$$

This leaves the case where e is of the form $e_1 \circ e_2$. Let n be the first node in preorder in the syntax tree of e that is not an application of $\circ$, and let e_3 be the expression rooted at n. By associativity of $\circ$, we can equivalently write e in the form $e_3 \circ e_4$, where e_4 equals the composition of all right-child expressions from the parent of n up to the root (in that order). Note that $e_3 \circ e_4$ has the same size as e. We now consider the different possibilities for the form of e_3:

$$\pi_1(id \circ e_4) = \pi_1(e_4)$$
$$\pi_1(di \circ e_4) = \pi_1(di \circ \pi_1(e_4))$$
$$\pi_1(R \circ e_4) = \pi_1(R \circ \pi_1(e_4))$$
$$\pi_1(R^{-1} \circ e_4) = \pi_2(\pi_1(e_4) \circ R)$$
$$\pi_1(\pi_j(e_5) \circ e_4) = \pi_j(e_5) \circ \pi_1(e_4)$$
$$\pi_1(\overline{\pi}_j(e_5) \circ e_4) = \overline{\pi}_j(e_5) \circ \pi_1(e_4)$$
$$\pi_1((e_5 \cup e_6) \circ e_4) = \pi_1(e_5 \circ e_4) \cup \pi_1(e_6 \circ e_4)$$
$$\overline{\pi}_1(id \circ e_4) = \overline{\pi}_1(e_4)$$
$$\overline{\pi}_1(di \circ e_4) = \overline{\pi}_1(di \circ \pi_1(e_4))$$
$$\overline{\pi}_1(R \circ e_4) = \overline{\pi}_1(R \circ \pi_1(e_4))$$
$$\overline{\pi}_1(R^{-1} \circ e_4) = \overline{\pi}_2(\pi_1(e_4) \circ R)$$
$$\overline{\pi}_1(\pi_j(e_5) \circ e_4) = \overline{\pi}_j(e_5) \cup \overline{\pi}_1(e_4)$$
$$\overline{\pi}_1(\overline{\pi}_j(e_5) \circ e_4) = \pi_j(e_5) \cup \overline{\pi}_1(e_4)$$
$$\overline{\pi}_1((e_5 \cup e_6) \circ e_4) = \overline{\pi}_1(e_5 \circ e_4) \circ \overline{\pi}_1(e_6 \circ e_4).$$

The crucial rules that eliminate inverse in the composition step are the fourth and the fourth-last. Hence we prove their correctness formally. Let G be an arbitrary graph. Then,

$$\begin{aligned}(x,x) \in \pi_1(R^{-1} \circ e_4)(G) &\iff \exists y : (x,y) \in R^{-1} \circ e_4(G)\\ &\iff \exists y \exists z : (x,z) \in R^{-1}(G) \wedge (z,y) \in e_4(G)\\ &\iff \exists z : (z,x) \in R(G) \wedge (z,z) \in \pi_1(e_4)(G)\\ &\iff \exists z : (z,x) \in \pi_1(e_4) \circ R(G)\\ &\iff (x,x) \in \pi_2(\pi_1(e_4) \circ R)(G).\end{aligned}$$

This proves the fourth rule. The fourth-last rule follows from the fourth rule and the fact that $\overline{\pi}_i(e') = id - \pi_i(e')$. This handles $\pi_1(e)$ and $\overline{\pi}_1(e)$.

To handle $\pi_2(e)$ and $\overline{\pi}_2(e)$, let n now be the first node in reverse preorder that is not an application of $\circ$. We can now write e as $e_4 \circ e_3$. The proof is now similar:

$$\pi_2(e_4 \circ id) = \pi_2(e_4)$$
$$\pi_2(e_4 \circ di) = \pi_2(\pi_2(e_4) \circ di)$$
$$\pi_2(e_4 \circ R) = \pi_2(\pi_2(e_4) \circ R)$$
$$\pi_2(e_4 \circ R^{-1}) = \pi_1(R \circ \pi_2(e_4))$$
$$\pi_2(e_4 \circ \pi_j(e_5)) = \pi_2(e_4) \circ \pi_j(e_5)$$
$$\pi_2(e_4 \circ \overline{\pi}_j(e_5)) = \pi_2(e_4) \circ \overline{\pi}_j(e_5)$$
$$\pi_2(e_4 \circ (e_5 \cup e_6)) = \pi_2(e_4 \circ e_5) \cup \pi_2(e_4 \circ e_6)$$
$$\overline{\pi}_2(e_4 \circ id) = \overline{\pi}_2(e_4)$$
$$\overline{\pi}_2(e_4 \circ di) = \overline{\pi}_2(\pi_2(e_4) \circ di)$$
$$\overline{\pi}_2(e_4 \circ R) = \overline{\pi}_2(\pi_2(e_4) \circ R)$$
$$\overline{\pi}_2(e_4 \circ R^{-1}) = \overline{\pi}_1(R \circ \pi_2(e_4))$$
$$\overline{\pi}_2(e_4 \circ \pi_j(e_5)) = \overline{\pi}_j(e_5) \cup \overline{\pi}_2(e_4)$$
$$\overline{\pi}_2(e_4 \circ \overline{\pi}_j(e_5)) = \pi_j(e_5) \cup \overline{\pi}_2(e_4)$$
$$\overline{\pi}_2(e_4 \circ (e_5 \cup e_6)) = \overline{\pi}_2(e_4 \circ e_5) \circ \overline{\pi}_2(e_4 \circ e_6).$$

In particular, if e is an expression in $\mathcal{N}(F \cup \{^{-1}\})$, it follows from the above that $\pi_1(e)$ is expressible in $\mathcal{N}(F \cup \{\pi\})$. Proposition 4.2 now follows from the observation that, for any graph G, $e(G)$ is nonempty if and only if $\pi_1(e)(G)$ is nonempty. □

Remark 4.4. Proposition 4.2 may remind one of a similar result known for XPath on trees [7,28,37] where downward XPath is known to be as powerful as full XPath for queries evaluated at the root. However, an important difference is that we are using projections both on the first and second column of a relation, whereas in the result on trees only the first projection is present.

Indeed, Proposition 4.2 no longer holds for a language which only contains the first, but not the second projection, or vice versa. Consider the following two graphs $G_1 = \{R(a,b), S(c,b)\}$ en $G_2 = \{R(a,b), S(c,d)\}$. For any expression $e \in \mathcal{N}(\pi_1)$ it must be that $e(G_1) \subseteq \{(a,a),(b,b),(c,c),(a,b),(c,b)\}$. It is not hard to see that for each $(x,y) \in \{(a,a),(b,b),(c,c),(a,b)\}$, $(x,y) \in e(G_1)$ iff $(x,y) \in e(G_2)$ and $(c,b) \in e(G_1)$ iff $(c,d) \in e(G_2)$. Therefore, it is clear that G_1 and G_2 are indistinguishable in $\mathcal{N}(\pi_1)$. They are, however, distinguishable in $\mathcal{N}(^{-1})$ by $R \circ S^{-1}$.

Remark 4.5. Notice that the translation used to eliminate converse in the proof of Proposition 4.2 could blow-up the size of the expressions exponentially. Indeed, define a family of expressions inductively as follows: $e_0 = T$ and $e_{n+1} = \pi_1((R \cup T) \circ e_n)$. Let us denote the size of an expression e as $|e|$. Clearly, $|e_0| = 0$ and $|e_{n+1}| = |e_n| + 5$, which implies that $|e_n|$ is linear in n. Now, let e'_n be the expression formed from e_n according to the rules outlined in the proof of Proposition 4.2. Clearly, $e'_0 = T$ and $e'_{n+1} = \pi_1(R \circ e'_n) \cup \pi_1(S \circ e'_n)$. Therefore, $|e'_0| = 1$ and $|e'_{n+1}| = 2|e'_n| + 7$, which implies that $|e'_n| \geqslant 2^n$.

On the other hand, our translation is never worse than single-exponential. We leave open whether a polynomial translation is possible. Interestingly, the analogous question about the complexity of translating from FO^3 to $\mathcal{N}(di, ^{-1}, -)$, mentioned in the Introduction, has not yet been addressed in the literature. For fragments of FO^2, a relevant result has been reported [11].

4.1. Proof of Proposition 4.1

We begin by recalling some basic terminology and notions concerning conjunctive queries [4]. A *conjunctive query with nonequalities* is expressed in the form $H \leftarrow B$. Here the body B is a finite set of relation atoms over the vocabulary Λ, as well as nonequalities of the form $x \neq y$. The head H is a tuple of variables from B. The head may be the empty tuple in which case a boolean query is expressed.

Given a conjunctive query $Q : H \leftarrow B$ and a graph G, an *assignment* is a function f from the set of variables in Q to $\mathrm{adom}(G)$. We call f a *matching* of B in G if for each relation atom $R(x,y)$ in B, we have $(f(x), f(y)) \in R(G)$, and for each $x \neq y$ in B we have $f(x) \neq f(y)$. The evaluation of Q on G is then defined as

$$Q(G) = \{f(H) \mid f \text{ is a matching from } B \text{ to } G\}.$$

In particular, if H is empty then $Q(G)$ is either $\{()\}$ or empty; these two possible results are interpreted as the boolean values *true* and *false* respectively.

A query Q_1 is said to be *contained* in a query Q_2, if for every graph G we have $Q_1(G) \subseteq Q_2(G)$. This is denoted by $Q_1 \subseteq Q_2$.

If B is the body of a conjunctive query with nonequalities, then B^{rel} denotes the set of relation atoms in B. As is customary in the theory of conjunctive queries, we can view the body of a conjunctive query without nonequalities as a graph whose nodes are the variables.

Recall that a homomorphism is a matching from a body without nonequalities to another body without nonequalities, viewed as a graph.

Lemma 4.6. *Let $Q_1 : H_1 \leftarrow B_1$ and $Q_2 : H_2 \leftarrow B_2$ be conjunctive queries with nonequalities. If $Q_1 \subseteq Q_2$ then there exists a homomorphism $h : B_2^{\text{rel}} \rightarrow B_1^{\text{rel}}$.*

Proof. Notice that $H_1 \in Q_1(B_1^{\text{rel}})$ since the identity map is clearly a matching. Hence $H_1 \in Q_2(B_1^{\text{rel}})$ because $Q_1 \subseteq Q_2$ by hypothesis. Therefore there exists a matching $f : B_2 \rightarrow B_1^{\text{rel}}$, which is also a matching from B_2^{rel} to B_1^{rel}, and is hence the desired homomorphism. □

We say that a directed graph G is a *chain* if it has no loops or cycles and its *undirected* version is isomorphic to the undirected chain with nodes $1, \ldots, n$ where n is the number of nodes of G. Such a chain has edges $\{i, i+1\}$ for $i = 1, \ldots, n-1$. Beware that in this terminology, a chain may have forward as well as backward edges, as illustrated in Fig. 1.

The following lemma can easily be proven by structural induction.

Lemma 4.7. *If e is a union-free expression in $\mathcal{N}(^{-1}, di)$, then there exists an equivalent conjunctive query $Q : H(x, y) \leftarrow B$ with nonequalities such that B^{rel} has the form of a disjoint union of chains.*

Let Q_{ZZZ} be the conjunctive query $() \leftarrow B_{ZZZ}$ that checks for the existence of the pattern displayed in Fig. 2. The name ZZZ is derived from the characteristic triple zigzag form of the pattern. For later use, we show the following. (Recall that an *endomorphism* of a structure A is a homomorphism from A to itself.)

Lemma 4.8. *The B_{ZZZ} pattern has no endomorphism except for the identity.*

Proof. Let f be an endomorphism of the B_{ZZZ} pattern in Fig. 2. We first show that $f(a) = a$. Note that there has to start a directed path of length 6 in $f(a)$ for the homomorphism property to hold since there starts a directed path of length 6 in a. Therefore $f(a) = a$ or $f(a) = j$. If $f(a) = j$ then $f(g) = k$, and hence $f(j) = l$. This, however, is not possible since there starts a directed path of length 6 in j but not in l. Therefore $f(a) = a$.

Now, the only thing left to verify is that no chain starting in a can be mapped homomorphically on another chain starting in a. First note that every chain starting in a has a very special structure, i.e., a path of forward edges, followed by an inverted edge, which is again followed by the same number of forward edges as before the inverted edge. Therefore, it is clear that a chain C_1 starting in a can only be mapped on another chain $C_2 \neq C_1$ starting in a, if and only if, the number of forward edges in C_1 minus one is at most the number of forward edges in C_2 preceding the inverted edge. In our graph, however, the number of forward edges in every chain starting in a minus one is at least seven, and the number of forward edges in every chain starting in a preceding the inverted edge is at most six. Therefore we can conclude that f maps every node onto itself as desired. □

We are now ready to prove Proposition 4.1.

Proof of Proposition 4.1. The boolean query Q_{ZZZ} is expressible in $\mathcal{N}(^{-1}, \pi)$ by

$$\pi_1(R^4 \circ R^{-1} \circ R^4) \circ \pi_1(R^5 \circ R^{-1} \circ R^5) \circ \pi_1(R^6 \circ R^{-1} \circ R^6).$$

This can be seen to be equivalent to

$$\pi_1(R^4 \circ \pi_2(\pi_1(R^4) \circ R)) \circ \pi_1(R^5 \circ \pi_2(\pi_1(R^5) \circ R)) \circ \pi_1(R^6 \circ \pi_2(\pi_1(R^6) \circ R))$$

in $\mathcal{N}(\pi)$ (a general argument for a result of this type will be given in the proof of Proposition 4.2). Let us now, for the sake of contradiction, assume that Q_{ZZZ} is also expressible in $\mathcal{N}(^{-1}, di)$ by an expression Q. Hence, for every graph G: (1) if $Q_{ZZZ}(G) = \mathit{true}$ then $Q(G) \neq \emptyset$, and (2) if $Q(G) \neq \emptyset$ then $Q_{ZZZ}(G) = \mathit{true}$. Since unions in $\mathcal{N}(^{-1}, di)$ can always be brought outside, we can assume that $Q = \bigcup_{i=0}^{n} e_i$ for some $n \in \mathbb{N}$ where each e_i is a union-free expression in $\mathcal{N}(^{-1}, di)$. Now, since $Q_{ZZZ}(B_{ZZZ}) = \mathit{true}$, we also have $Q(B_{ZZZ}) = \cup_{i=0}^{n} e_i(B_{ZZZ}) \neq \emptyset$. Hence there exists $e \in \{e_0, \ldots, e_n\}$ such that $e(B_{ZZZ}) \neq \emptyset$. By Lemma 4.7, e is equivalent to a conjunctive query with nonequalities $H_e \leftarrow B_e$ such that B_e^{rel} is a disjoint union of chains. Furthermore, since $e(B_{ZZZ}) \neq \emptyset$ there exists a matching $f : B_e^{\text{rel}} \rightarrow B_{ZZZ}$ which is a homomorphism by definition.

Now let Q_e be the conjunctive query with nonequalities $() \leftarrow B_e$ so that $Q_e(G) = \mathit{true}$ if and only if $e(G) \neq \emptyset$ for every graph G. Since $e(G) \subseteq Q(G)$ for any graph G, $Q_e(G) = \mathit{true}$ implies $Q(G) \neq \emptyset$, whence by (2) $Q_{ZZZ}(G) = \mathit{true}$. Therefore $Q_e \subseteq Q_{ZZZ}$. By Lemma 4.6 there is a homomorphism g from B_{ZZZ} into B_e^{rel}. Notice that in the B_{ZZZ} pattern displayed in Fig. 2, the left most node, labeled a, has three outgoing edges. Furthermore, since B_e^{rel} is a disjoint union of chains, no node in B_e^{rel} has 3 outgoing edges, and hence two out of $g(b), g(c)$ and $g(d)$ are equal. Thus g is not injective.

Now consider g followed by f. This function is an endomorphism of B_{ZZZ}. Because g is not injective, this endomorphism is not injective, and hence certainly not the identity, which contradicts Lemma 4.8. Therefore Q does not exist. □

Fig. 1. Example of a chain.

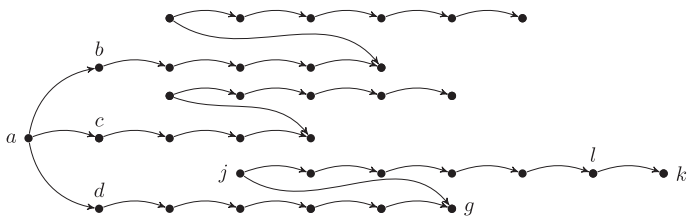

Fig. 2. Query pattern B_{ZZZ} used to prove Proposition 4.1. All edges are assumed to have the same label *R*.

5. Path queries

In this section, we characterize the order $\leqslant^{\text{path}}$ of relative expressiveness for path queries by Theorem 5.2 below. Towards the statement of this characterization, first notice the following interdependencies between features:

$$\pi_1(e) = (e \circ e^{-1}) \cap id = (e \circ (id \cup di)) \cap id = \overline{\pi}_1(\overline{\pi}_1(e));$$
$$\pi_2(e) = (e^{-1} \circ e) \cap id = ((id \cup di) \circ e) \cap id = \overline{\pi}_2(\overline{\pi}_2(e));$$
$$\overline{\pi}_1(e) = id - \pi_1(e);$$
$$\overline{\pi}_2(e) = id - \pi_2(e);$$
$$e_1 \cap e_2 = e_1 - (e_1 - e_2).$$

Notice that these rewriting rules with *e* as their input variable provide a means to translate an expression into an equivalent expression in another language.

Inspired by the above interdependencies, for any set of nonbasic features *F*, we define $\overline{F}$ to be the smallest superset of *F* satisfying the following rules:

- If $\overline{\pi} \in \overline{F}$, then $\pi \in \overline{F}$;
- If $\cap \in \overline{F}$ and $di \in \overline{F}$, then $\pi \in \overline{F}$;
- If $\cap \in \overline{F}$ and $^{-1} \in \overline{F}$, then $\pi \in \overline{F}$;
- If $- \in \overline{F}$ and $\pi \in \overline{F}$, then $\overline{\pi} \in \overline{F}$.
- If $- \in \overline{F}$, then $\cap \in \overline{F}$;

We can compute $\overline{F}$ from *F* by repeated application of the above rules, a process which terminates quickly after at most three iterations. For example, $\overline{\{-, ^{-1}\}} = \{-, ^{-1}, \cap, \pi, \overline{\pi}\}$.

Notice that, if $F_1 \subseteq \overline{F}_2$, we can always rewrite an expression $e \in \mathcal{N}(F_1)$ into an equivalent expression in $\mathcal{N}(F_2)$ using the rewriting rules displayed above. Notice that Therefore, we obtain

Proposition 5.1. *If* $F_1 \subseteq \overline{F}_2$, *then* $\mathcal{N}(F_1) \leqslant^{\text{path}} \mathcal{N}(F_2)$.

We will actually show that the converse also holds, whence

Theorem 5.2. $\mathcal{N}(F_1) \leqslant^{\text{path}} \mathcal{N}(F_2)$ *if and only if* $F_1 \subseteq \overline{F}_2$.

The "only if" direction of Theorem 5.2 requires a detailed analysis. For clarity of presentation, we divide the languages under consideration into two classes, i.e., the class $\mathcal{C}$ of languages without intersection, and the class $\mathcal{C}[\cap]$ of languages with intersection. Formally:

$$\mathcal{C} = \{\mathcal{N}(F) | \cap \notin \overline{F}\},$$
$$\mathcal{C}[\cap] = \{\mathcal{N}(F) | \cap \in \overline{F}\}.$$

We first establish the "only if" direction for the cases where $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ belong to the same class. We do so for each class separately in Sections 5.1 and 5.2. Finally, in Section 5.3, we consider the case where $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ belong to distinct classes.

5.1. Languages without $\cap$

In this subsection, we show the "only if" direction of Theorem 5.2, restricted to $\mathcal{C}$, the class of languages without $\cap$. Stated positively, the proposition states that for fragments F_1 and F_2 using only the operators di, π and $\overline{\pi}$, $\mathcal{N}(F_1) \leqslant^{\text{path}} \mathcal{N}(F_2)$ can only hold if $F_1 \subseteq \overline{F_2}$.

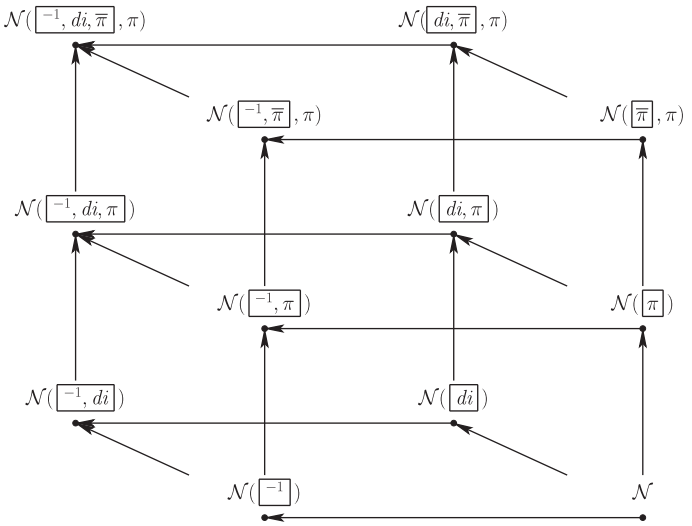

Fig. 3. The Hasse diagram of $\leqslant^{\text{path}}$ for $\mathcal{C}$. For each language, the boxed features are a minimal set of nonbasic features defining the language, while the other features can be derived from them in the sense of Theorem 5.2 (using the appropriate interdependencies).

Proposition 5.3. *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be in $\mathcal{C}$. If $F_1 \not\subseteq \overline{F}_2$, then $\mathcal{N}(F_1) \not\leqslant^{\text{path}} \mathcal{N}(F_2)$.*

Propositions 5.1 and 5.3 combined yield the Hasse diagram of $\leqslant^{\text{path}}$ for $\mathcal{C}$, shown in Fig. 3. It is indeed readily verified that for any two languages $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ in $\mathcal{C}$, there is a path from $\mathcal{N}(F_1)$ to $\mathcal{N}(F_2)$ in Fig. 3 if and only if $F_1 \subseteq \overline{F}_2$.

Towards a proof of Proposition 5.3, we first establish an auxiliary proposition. For later use, we sometimes prove results that are stronger than strictly needed for this purpose.

Proposition 5.4. *Let F_1 and F_2 be sets of nonbasic features.*

1. If $di \in \overline{F}_1$ and $di \notin \overline{F}_2$, then $\mathcal{N}(F_1) \not\leqslant^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.
2. If $\overline{\pi} \in \overline{F}_1, \overline{\pi} \notin \overline{F}_2$, and $- \notin \overline{F}_2$, then $\mathcal{N}(F_1) \not\leqslant^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.
3. If $^{-1} \in \overline{F}_1$ and $^{-1} \notin \overline{F}_2$, then $\mathcal{N}(F_1) \not\leqslant^{\text{path}}_{\text{strong}} \mathcal{N}(F_2)$.
4. If $\pi \in \overline{F}_1$ and $F_2 \subseteq \{^{-1}, di\}$, then $\mathcal{N}(F_1) \not\leqslant^{\text{bool}} \mathcal{N}(F_2)$.

Proof. For (1), consider a graph G_1 consisting of two self-loops, and a graph G_2 consisting of a single self-loop, all with the same label. For any nontrivial expression e not using $di, -$ or $\overline{\pi}$, it is evident that $e(G_1)$ and $e(G_2)$ both contain all possible self-loops in G_1 and G_2 respectively. Therefore, applying $-$ or $\overline{\pi}$ to any such expressions leads to expressions that show similar behavior on G_1 and G_2. More specifically, they select either all self-loops in both G_1 and G_2, or select nothing in both graphs simultaneously. The same reasoning can now be applied to general expressions in $\mathcal{N}(F_2)$. This reasoning shows that G_1 and G_2 cannot be distinguished in $\mathcal{N}(F_2)$. They are, however, distinguishable by in $\mathcal{N}(F_1)$ by $di \neq \emptyset$.

For (2), notice that $\overline{F_2} \subseteq \{di, \pi, \cap, ^{-1}, ^{+}\}$, whence $\mathcal{N}(F_2)$ only contains monotone expressions. Therefore it is clear that a non-monotone query such as $\overline{\pi}_2(R) \neq \emptyset$ is not expressible in $\mathcal{N}(F_2)$.

For (3), we establish strong separation at the level of path queries as explained in Section 3.1. Thereto, we consider the graph G shown in Fig. 5. By the Brute-Force method described in Section 3.2.1. we can exhaustively enumerate all the possible result relations $e(G)$ for all expressions $e \in \mathcal{N}(di, -, ^{+})$, i.e., not using converse. There are 128 relations in this list. It can then be verified that G^{-1} is not present in the list.[4]

The proof of (4) follows directly from Proposition 4.1 since $\mathcal{N}(\pi) \leqslant^{\text{path}} \mathcal{N}(F_1)$ and $\mathcal{N}(F_2) \leqslant^{\text{path}} \mathcal{N}(^{-1}, di)$. □

Proposition 5.4 is now used to show that for every pair F_1 and F_2 of sets of nonbasic features for which $F_1 \not\subseteq \overline{F}_2$ (i.e., for which there is no path in Fig. 3), that $\mathcal{N}(F_1) \not\leqslant^{\text{path}} \mathcal{N}(F_2)$. The remainder of the proof of Proposition 5.3 is a combinatorial analysis to verify that Proposition 5.4 covers all the cases.

Proof of Proposition 5.3. First, suppose that $\overline{\pi} \in F_2$. Then, $F_1 \not\subseteq \overline{F_2}$ if and only if $F_1 \cap \{di, ^{-1}\} \not\subseteq F_2 \cap \{di, ^{-1}\}$. Hence we have the following possible scenarios: $di \in F_1$ and $di \notin \overline{F_2}$; or $^{-1} \in F_1$ and $^{-1} \notin \overline{F_2}$. If $di \in F_1$ and $di \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leqslant^{\text{path}} \mathcal{N}(F_2)$ due to Proposition 5.4(1). Otherwise, we achieved the result due to Proposition 5.4(3).

[4] Note that if we would have used a simpler graph G, say G consisting of a single edge, then G^{-1} would be expressible without using converse, using the expression $di - R$.

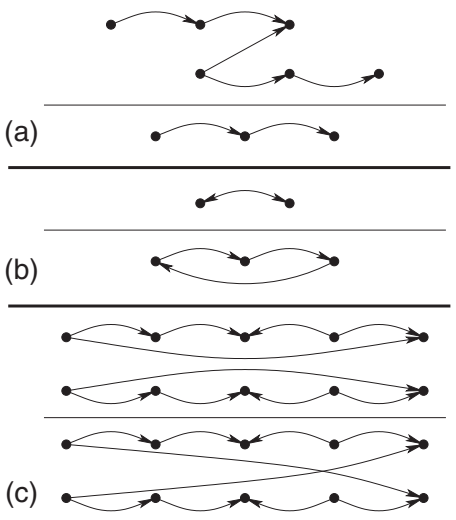

Fig. 4. Graph pairs used to prove $\not\leqslant^{\text{bool}}_{\text{strong}}$ results in Sections 5 and 6. All edges are assumed to have the same label R.

On the other hand, suppose that $\overline{\pi} \notin F_2$. Then, $F_2 = \overline{F_2}$. Thus, $F_1 \not\subseteq \overline{F_2}$ if and only if $F_1 \not\subseteq F_2$. Hence there has to exists some $x \in F_1$ such that $x \notin F_2$. Furthermore, since $F_1 \subseteq \{di, \pi, \overline{\pi}, ^{-1}\}$ and $F_2 \subseteq \{di, \pi, ^{-1}\}$ Proposition 5.4 can be applied. Notice that we cannot apply this proposition directly since it makes use of $\overline{F_1}$ instead of F_1. This, however, is no issue since $F_1 \subseteq \overline{F_1}$. □

5.2. Languages with ∩

In this subsection, we show the "only if" direction of Theorem 5.2, restricted to $\mathcal{C}[\cap]$, the class of languages with ∩.

Proposition 5.5. *Let both $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be in $\mathcal{C}[\cap]$. If $F_1 \not\subseteq \overline{F}_2$, then $\mathcal{N}(F_1) \not\leqslant^{\text{path}} \mathcal{N}(F_2)$.*

Propositions 5.1 and 5.5 combined yield the Hasse diagram of $\leqslant^{\text{path}}$ for $\mathcal{C}[\cap]$, shown in Fig. 6.
Towards a proof of Proposition 5.5, we first establish the following.

Proposition 5.6. *Let F_1 and F_2 be sets of nonbasic features.*

1. If $- \in \overline{F}_1$ and $- \notin \overline{F}_2$, then $\mathcal{N}(F_1) \not\leqslant^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.
2. If $\pi \in \overline{F}_1$, and $F_2 \subseteq \{-, \cap\}$, then $\mathcal{N}(F_1) \not\leqslant^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.

Proof. For (1), consider a 3-clique G_1, and a bow-tie G_2 consisting of two 3-cliques (both graphs contain a self-loop on every node). It can be proven by straightforward induction and case analysis that for any nontrivial expression $e \in \mathcal{N}(di, ^{-1}, \cap, ^{+})$ at least $id(G_i) \subseteq e(G_i)$ or $R - id(G_i) \subseteq e(G_i)$. In either case it is clear that a projection of any nontrivial expression in $\mathcal{N}(di, ^{-1}, \cap, ^{+})$ evaluated on both graphs leads to all self-loops. Using this fact, it can be seen that a coprojection of any expression in $\mathcal{N}(di, ^{-1}, \cap, \overline{\pi}, \pi, ^{+})$ leads to either all self-loops, or a completely empty query result on both graphs simultaneously. Therefore, no expression in $\mathcal{N}(di, ^{-1}, \cap, \overline{\pi}, \pi, ^{+})$ can distinguish G_1 and G_2. The graphs, however, are distinguishable by the boolean query expressed by $R^2 - R$.

For (2), consider the graphs displayed in Fig. 4 (a). Notice that expressions in $\mathcal{N}$ select paths of the same length in both graphs simultaneously, e.g., if an expression selects all paths of length two in one graph, it also selects all the paths of length two in the other and vice versa. Therefore, expressions using set difference evaluate to empty or nonempty on both graphs simultaneously. Thus, expressions in $\mathcal{N}(-)$ cannot distinguish the considered graphs, whence they are indistinguishable in $\mathcal{N}(F_2)$ as well since $\mathcal{N}(F_2) \leqslant^{\text{bool}} \mathcal{N}(-)$. The graphs, however, are distinguishable in $\mathcal{N}(F_1)$ by the boolean query expressed by $\pi_1(R^2) \circ R \circ \pi_2(R^2)$. □

Propositions 5.4 and 5.6 are now used to show that for every pair F_1 and F_2 of sets of nonbasic features for which $F_1 \not\subseteq \overline{F}_2$ (i.e., for which there is no path in Fig. 6), that $\mathcal{N}(F_1) \not\leqslant^{\text{path}} \mathcal{N}(F_2)$.

Fig. 5. Graph used to prove Proposition 5.4 (3). Both edges are assumed to have the same label R.

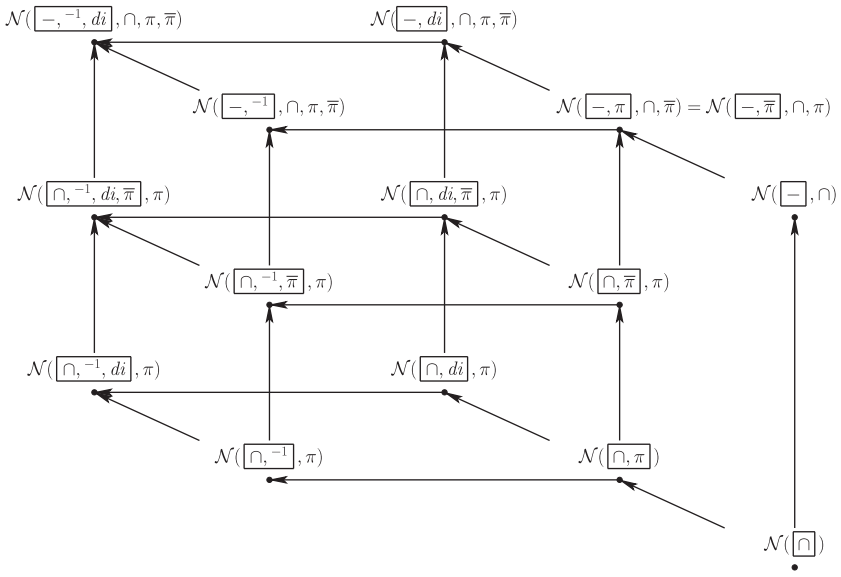

Fig. 6. The Hasse diagram of $\leqslant^{\text{path}}$ and $\leqslant^{\text{bool}}$ for $\mathcal{C}[\cap]$.

The remainder of the proof of Proposition 5.5 is a combinatorial analysis to verify that Propositions 5.4 and 5.6 cover all relevant cases.

Proof of Proposition 5.5. By definition $\cap \in \overline{F_1}$ and $\cap \in \overline{F_2}$ since both $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ are in $\mathcal{C}[\cap]$. Hence, $F_1 \not\subseteq \overline{F_2}$ if and only if there exists $x \in \{\pi, \overline{\pi}, di, ^{-1}, -\}$ such that $x \in F_1$ and $x \notin \overline{F_2}$. We will consider every such x and show that our result directly follows from Propositions 5.4 or 5.6.

If $x = di, x = ^{-1}$ or $x = -$, then respectively Proposition 5.4(1), (3) or 5.6(1) gives us the desired result.

If $x = \pi$, then clearly $\pi \notin \overline{F_2}$ if and only if $F_2 \cap \{di, ^{-1}, \overline{\pi}, \pi\} = \emptyset$. Hence $F_2 \subseteq \{\cap, -\}$. Now, we can apply Proposition 5.6(2), which proves the result.

If $x = \overline{\pi}$, then using the interdependencies introduced in the beginning of Section 5 we get

$$\overline{\pi} \notin \overline{F_2} \iff - \notin F_2 \vee (- \in F_2 \wedge \pi \notin \overline{F_2}).$$

So we have two scenarios. If $- \notin F_2$ then we can apply Proposition 5.4(2) to prove our result. On the other hand, when $- \in F_2$ we cannot apply Proposition 5.4(2). As said above, now π cannot be in $\overline{F_2}$. Furthermore, note that in this scenario

$$- \in F_2 \wedge \pi \notin \overline{F_2} \iff F_2 \cap \{^{-1}, di\} = \emptyset,$$

which implies that $F_2 \subseteq \{\cap, -\}$. Moreover, $\pi \in \overline{F_1}$ since $\overline{\pi} \in \overline{F_1}$. Hence, we can apply proposition 5.6(2), which proves the result. □

5.3. Cross-relationships between subdiagrams

To finish the proof of Theorem 5.2, we finally show the "only if" direction for the case where $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ belong to different classes.

Proposition 5.7. *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be languages such that one language belongs to $\mathcal{C}$, and the other language belongs to $\mathcal{C}[\cap]$. If $F_1 \not\subseteq \overline{F_2}$, then $\mathcal{N}(F_1) \not\leqslant^{\text{path}} \mathcal{N}(F_2)$.*

Towards a proof of Proposition 5.7, we first establish the following.

Proposition 5.8. *Let F_1 and F_2 be sets of nonbasic features. If $\cap \in \overline{F_1}$ and $\cap \notin \overline{F_2}$, then $\mathcal{N}(F_1) \not\leqslant^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

Proof. Since $\cap \notin \overline{F_2}$ it must be that $F_2 \subseteq \{di, ^{-1}, \pi, \overline{\pi}, ^{+}\}$. So, it is sufficient to find a boolean query expressible in $\mathcal{N}(F_1)$, which is not expressible in $\mathcal{N}(di, ^{-1}, \overline{\pi}, ^{+})$. Consider the graphs G_1 and G_2 in Fig. 4 (b). Notice that there starts and ends a path of every length in each node in both graphs. Utilizing this fact, it can be shown that for any nontrivial expression $e \in \mathcal{N}(di, ^{-1}, ^{+})$, it must be that $\pi_i(e)(G_j) = id(G_j)$. Using this, it can be seen that the coprojection of any expression in $\mathcal{N}(di, ^{-1}, \overline{\pi}, ^{+})$ leads to either all self-loops, or a completely empty query result on both graphs simultaneously. Therefore, no expression in $\mathcal{N}(di, ^{-1}, \overline{\pi}, ^{+})$ can distinguish G_1 and G_2. The graphs, however, are distinguishable by the boolean query expressed by $R^2 \cap id$. □

As detailed below, Propositions 5.4, 5.6 and 5.8 are now subsequently used to show that for every pair F_1 and F_2 of sets of nonbasic features for which $F_1 \not\subseteq \overline{F_2}$, that $\mathcal{N}(F_1) \not\leqslant^{\text{path}} \mathcal{N}(F_2)$, in the same way as in Sections 5.1 and 5.2.

The remainder of the proof of Proposition 5.7 is again a combinatorial analysis to verify that the above-mentioned propositions cover all relevant cases.

Proof of Proposition 5.7. First, suppose that $\mathcal{N}(F_1) \in \mathcal{C}[\cap]$ and $\mathcal{N}(F_2) \in \mathcal{C}$. Then, by definition $\cap \in \overline{F_1}$ and $\cap \notin \overline{F_2}$. The result now follows directly from Proposition 5.8.

On the other hand, suppose that $\mathcal{N}(F_1)$ is in $\mathcal{C}$ and $\mathcal{N}(F_2)$ is in $\mathcal{C}[\cap]$. Clearly, then $F_1 \not\subseteq \overline{F_2}$ if and only if $F_1 \not\subseteq \overline{F_2} - \{\cap, -\}$. Hence at least one feature x of $di, \pi, \overline{\pi}, {}^{-1}$ is present in F_1 but missing in $\overline{F_2}$. We will consider every such x and show that our result directly follows from Propositions 5.4, or 5.6.

If $x = di$ or $x = {}^{-1}$, then respectively Proposition 5.4(1) or (3) gives us the desired result.

If $x = \pi$ then $\overline{\pi} \notin F_2$ by the interdependencies introduced in the beginning of Section 5. Furthermore, $\overline{F_2} \cap \{{}^{-1}, di\} = \emptyset$ since by hypothesis $\cap \in \overline{F_2}$. Therefore $F_2 \subseteq \{-, \cap\}$, and hence Proposition 5.6(2) can be applied, which proves the result.

If $x = \overline{\pi}$ then $\overline{F_2} \cap \{-, \pi\} \neq \{-, \pi\}$. Suppose that $- \notin \overline{F_2}$, then our result follows from Proposition 5.4(2). On the other hand, if $\pi \notin \overline{F_2}$, then the result follows from the previous case since $\pi \in \overline{F_1}$. □

Propositions 5.1, 5.3, 5.5 and 5.7, together prove Theorem 5.2.

Hence, the Hasse diagram of $\leqslant^{\text{path}}$ can be obtained from the subdiagrams for $\mathcal{C}$ and $\mathcal{C}[\cap]$ by simply adding the 12 canonical inclusion arrows between the subdiagram for $\mathcal{C}$ and the subdiagram for $\mathcal{C}[\cap]$. However, in the presence of $\cap, di$ or ${}^{-1}$ gives π, so the arrows from $\mathcal{N}(di)$ to $\mathcal{N}(\cap, di, \pi), \mathcal{N}({}^{-1})$ to $\mathcal{N}(\cap, {}^{-1}, \pi)$, and $\mathcal{N}({}^{-1}, di)$ to $\mathcal{N}(\cap, {}^{-1}, di, \pi)$ are transitive, and can therefore be omitted.

So, all paths between the subdiagrams are induced by these canonical inclusion arrows and the 5 equations from the beginning of Section 5.

6. Boolean queries

In this section, we characterize the order $\leqslant^{\text{bool}}$ of relative expressiveness for boolean queries by Theorem 6.1 below.

Towards the statement of this characterization, first observe that $\mathcal{N}(F_1) \leqslant^{\text{path}} \mathcal{N}(F_2)$ implies $\mathcal{N}(F_1) \leqslant^{\text{bool}} \mathcal{N}(F_2)$. The converse does not hold, however. Indeed, from Proposition 4.2, it follows that, e.g., $\mathcal{N}({}^{-1}) \leqslant^{\text{bool}} \mathcal{N}(\pi)$. From Theorem 5.2, however, we know that $\mathcal{N}({}^{-1}) \not\leqslant^{\text{path}} \mathcal{N}(\pi)$.

To accommodate the collapse of ${}^{-1}$ in our characterization of $\leqslant^{\text{bool}}$, we introduce some new notation. For a set of nonbasic features F, define $\widehat{F}$ as follows.

$$\widehat{F} = \begin{cases} (F - \{{}^{-1}\}) \cup \{\pi\}, & \text{if } {}^{-1} \in \overline{F}, \cap \notin \overline{F}, \\ F, & \text{otherwise.} \end{cases}$$

For example, $\{\widehat{di, {}^{-1}}\} = \{di, \pi\}$.

We will establish the following characterization.

Theorem 6.1. *Let F_1 and F_2 be sets of nonbasic features. Then, $\mathcal{N}(F_1) \leqslant^{\text{bool}} \mathcal{N}(F_2)$ if and only if $F_1 \subseteq \overline{F_2}$ or $\widehat{F_1} \subseteq \overline{F_2}$*

The "if" direction of Theorem 6.1 is shown by Proposition 5.1 (since $\leqslant^{\text{path}}$ implies $\leqslant^{\text{bool}}$) and Proposition 6.2.

Proposition 6.2. *If $\widehat{F_1} \subseteq \overline{F_2}$ then $\mathcal{N}(F_1) \leqslant^{\text{bool}} \mathcal{N}(F_2)$.*

Proof. We distinguish two cases. If $F_1 \subseteq \overline{F_2}$, then $\mathcal{N}(F_1) \leqslant^{\text{path}} \mathcal{N}(F_2)$, by Proposition 5.1, whence $\mathcal{N}(F_1) \leqslant^{\text{bool}} \mathcal{N}(F_2)$.

In the other case, ${}^{-1} \in \overline{F_1}$, and $\cap \notin \overline{F_1}$. Hence $\mathcal{N}(F_1) \leqslant^{\text{bool}} \mathcal{N}(F_1 - \{{}^{-1}\} \cup \{\pi\}) = \mathcal{N}(\widehat{F_1})$ by Proposition 4.2. Furthermore, $\mathcal{N}(\widehat{F_1}) \leqslant^{\text{path}} \mathcal{N}(F_2)$ since $\widehat{F_1} \subseteq \overline{F_2}$ by Proposition 5.1, whence $\mathcal{N}(\widehat{F_1}) \leqslant^{\text{bool}} \mathcal{N}(F_2)$. Now, by transitivity $\mathcal{N}(F_1) \leqslant^{\text{bool}} \mathcal{N}(F_2)$ as desired. □

The converse of this proposition does not hold in general, e.g., $\mathcal{N}({}^{-1}) \leqslant^{\text{bool}} \mathcal{N}({}^{-1}, -)$ but $\{\widehat{{}^{-1}}\} = \{\pi\} \not\subseteq \overline{\{{}^{-1}, -\}} = \{{}^{-1}, -, \cap\}$.

The "only if" direction of Theorem 6.1, requires a detailed analysis, which proceeds along the same lines as the analysis in Section 5. We first establish the "only if" direction for the cases where $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ belong to the same class among $\mathcal{C}$ and $\mathcal{C}[\cap]$, and then consider the case where $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ belong to distinct classes.

6.1. Languages without ∩

In this subsection, we show the "only if" direction of Theorem 6.1, restricted to $\mathcal{C}$, the class of languages without $\cap$.

Proposition 6.3. *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be in $\mathcal{C}$. If $F_1 \not\subseteq \overline{F_2}$ and $\widehat{F_1} \not\subseteq \overline{F_2}$, then $\mathcal{N}(F_1) \not\leqslant^{\text{bool}} \mathcal{N}(F_2)$.*

Propositions 5.1, 6.2 and 6.3 combined yield the Hasse diagram of $\leqslant^{\text{bool}}$ for $\mathcal{C}$, shown in Fig. 7. It is indeed readily verified that for any two languages $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ in $\mathcal{C}$, there is a path from $\mathcal{N}(F_1)$ to $\mathcal{N}(F_2)$ in Fig. 7 if and only if $F_1 \subseteq \overline{F_2}$ or $\widehat{F_1} \subseteq \overline{F_2}$.

Towards a proof of Proposition 6.3, we first establish the following.

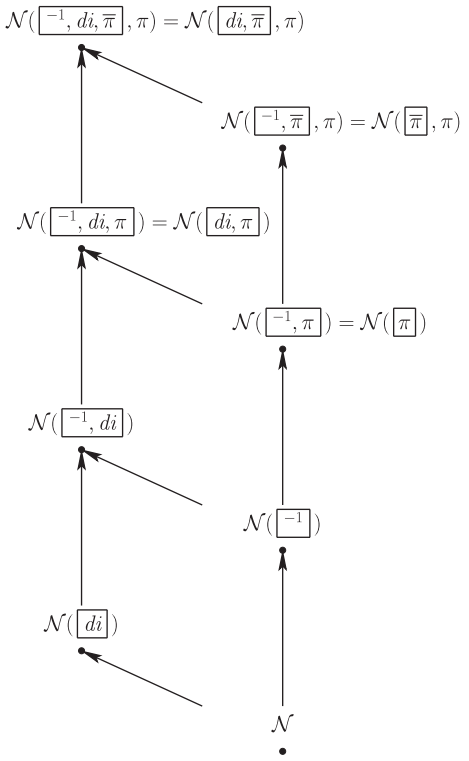

Fig. 7. The Hasse diagram of $\leqslant^{\text{bool}}$ for $\mathcal{C}$. For each language, the boxed features are a minimal set of nonbasic features defining the language, while the other features can be derived from them in the sense of Theorem 5.2 (using the appropriate interdependencies).

Proposition 6.4. *Let F be a set of nonbasic features. If $^{-1} \in \overline{F}$, then we have $\mathcal{N}(F) \not\leqslant^{\text{bool}}_{\text{strong}} \mathcal{N}(di)$.*

Proof. Let $\mathcal{C}$ be the class of all graphs G such that G is acyclic and adom(G) contains at least three elements, and let $e \in \mathcal{N}(di)$. We will show that on the class $\mathcal{C}$, the boolean query $e \neq \emptyset$ is either $\emptyset \neq \emptyset$ (always false) or equivalent to $R^m \neq \emptyset$ for some natural number m. Let us first show this for union-free expressions. Since $di^i = id \cup di$ in $\mathcal{C}$ for $i > 1$, we may assume that $e = R^{n_1} \circ di \circ R^{n_2} \circ di \circ \ldots \circ di \circ R^{n_k}$ where $k > 1$ and $n_1, \ldots, n_k$ are natural numbers greater than zero. We set m to be the maximum of the n_l for $1 \leqslant l \leqslant k$. Let G be an arbitrary graph in $\mathcal{C}$. Clearly, if $e(G) \neq \emptyset$ then $R^m(G) \neq \emptyset$. For the other direction, assume $(x, y) \in R^m(G)$. Since G is acyclic, $x \neq y$, so $(y, x) \in di(G)$. Hence $(x, x) \in R^m \circ di(G)$. For any $l \leqslant m$, we also have $(x, x) \in R^l \circ di(G)$. We conclude that $(x, x) \in R^{n_1} \circ di \circ \ldots \circ R^{n_k} \circ di(G)$. In particular, $e(G)$ is nonempty as desired.

For the claim to hold with union, it suffices to show it for a union of two union-free expressions. Indeed, the form R^m is union-free! So, consider an expression e of the form $R^{m_1} \cup R^{m_2}$. Then $e \neq \emptyset$ is equivalent to $R^{\min(m_1, m_2)} \neq \emptyset$, which proves the claim.

Now consider graphs G_1 and G_2 in Fig. 4(a). These graphs belong to $\mathcal{C}$, and are clearly indistinguishable by any expression of the form $R^m \neq \emptyset$. The graphs, however, are distinguishable by the boolean query $R^2 \circ R^{-1} \circ R^2 \neq \emptyset$. □

As detailed below, Propositions 5.4 and 6.4 are now subsequently used to show that for every pair F_1 and F_2 of sets of nonbasic features for which $F_1 \not\subseteq \widetilde{F}_2$, that $\mathcal{N}(F_1) \not\leqslant^{\text{bool}} \mathcal{N}(F_2)$, in the same way as in Sections 5.1 and 5.2.

The remainder of the Proposition 6.3 is again a combinatorial analysis to verify that the above-mentioned propositions cover all relevant cases.

Proof of Proposition 6.3. First, note that $F_1 \cup F_2 \subseteq \{^{-1}, \overline{\pi}, \pi, di\}$ since $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ are in $\mathcal{C}$. We will consider two cases: $\pi \in \overline{F_2}$ and $\pi \notin \overline{F_2}$. First we will consider $\pi \in \overline{F_2}$. Since $\widehat{F_1} \not\subseteq \overline{F_2}$, there must be another feature, not equal to $^{-1}$ or π present. If this feature is di, then Proposition 5.4(1) proves the result. On the other hand, if this feature is $\overline{\pi}$ then Proposition 5.4(2) proves the result.

Now consider the case where $\pi \notin \overline{F_2}$. Here, $\overline{\pi} \notin F_2$ and thus $F_1 \not\subseteq \overline{F_2} \subseteq \{di, ^{-1}\}$. Hence one of $^{-1}, \overline{\pi}, \pi$ or di is present in F_1 but missing in $\overline{F_2}$. If that feature is $^{-1}$, then $\overline{F_2} \subseteq \{di\}$, and hence Proposition 6.4 proves the result. On the other hand, if that feature is $\overline{\pi}, di$ or π, the result follows directly from Proposition 5.4. □

6.2. Languages with ∩

In this subsection, we show the "only if" direction of Theorem 6.1, restricted to $\mathcal{C}[\cap]$, the class of languages with ∩.

Proposition 6.5. *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be in $\mathcal{C}[\cap]$. If $F_1 \not\subseteq \overline{F_2}$ and $\widehat{F_1} \not\subseteq \overline{F_2}$, then $\mathcal{N}(F_1) \not\leqslant^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

Notice that since $\cap \in \overline{F_1}, \widehat{F_1} = F_1$. Hence, Theorem 5.2 and Proposition 6.5 combined show that $\leqslant^{\text{bool}}$ coincides with $\leqslant^{\text{path}}$ on $\mathcal{C}[\cap]$. As a result, the Hasse diagram of $\leqslant^{\text{bool}}$ for $\mathcal{C}[\cap]$ is the same as the Hasse diagram of $\leqslant^{\text{path}}$ for $\mathcal{C}[\cap]$ shown in Fig. 6. Note that, in addition, all separations are strong.

Towards a proof of Proposition 6.5, we first establish the following.

Proposition 6.6. *Let F_1 and F_2 be sets of nonbasic features. If $^{-1} \in F_1, \cap \in F_1$, and $^{-1} \notin F_2$, then $\mathcal{N}(F_1) \not\leqslant^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

Proof. The graphs G_1 and G_2 shown in Fig. 4 (c), top and bottom, are distinguished by the boolean query q expressed by $(R^2 \circ R^{-1} \circ R) \cap R$. On these graphs, the Brute-Force Algorithm of Section 3.2.1 does not terminate in a reasonable time. It can be verified in polynomial time, however, that for each pair $(a_1, b_1) \in \text{adom}(G_1)^2$, there exists $(a_2, b_2) \in \text{adom}(G_2)^2$ such that $(G_1, a_1, b_1) \simeq_k (G_2, a_2, b_2)$ for any depth k [15]. From Proposition 3.4, it follows that q is not expressible in $\mathcal{N}(F_2)$. □

The remainder of the proof of Proposition 6.5 proceeds as the proof of Proposition 5.5, except that Proposition 6.6 is used instead of Proposition 5.4 (3).

6.3. Cross-relationships between subdiagrams

To finish the proof of Theorem 6.1, we finally show the "only if" direction for the case where $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ belong to different classes.

Proposition 6.7. *Let $\mathcal{N}(F_1)$ and $\mathcal{N}(F_2)$ be languages such that one language belongs to $\mathcal{C}$, and the other language belongs to $\mathcal{C}[\cap]$. If $F_1 \not\subseteq \overline{F_2}$ and $\widehat{F_1} \not\subseteq \overline{F_2}$, then $\mathcal{N}(F_1) \not\leqslant^{\text{bool}} \mathcal{N}(F_2)$.*

Towards a proof of Proposition 6.7, we first establish the following.

Proposition 6.8. *Let F_1 be a set of nonbasic features. If $^{-1} \in \overline{F_1}$, and $F_2 \subseteq \{-, \cap\}$, then $\mathcal{N}(F_1) \not\leqslant^{\text{bool}}_{\text{strong}} \mathcal{N}(F_2)$.*

Proof. Consider the graphs G_1 and G_2 displayed in Fig. 4 (a) and define $R^0(G_i)$ to equal $id(G_i)$ for $i = 1, 2$. First, notice that $id(G_i), R(G_i)$, and $R^2(G_i)$ are pairwise disjoint for $i = 1, 2$. Utilizing this, it can be proven by straightforward induction that for every $e \in \mathcal{N}(-)$ there exists $Z \subseteq \{0, 1, 2\}$ such that $e(G_1) = \cup_{i \in Z} R^i(G_1)$ and $e(G_2) = \cup_{i \in Z} R^i(G_2)$. This clearly implies that G_1 and G_2 are indistinguishable in $\mathcal{N}(-)$. whence they are also indistinguishable in $\mathcal{N}(F_2)$ as well since $\mathcal{N}(F_2) \leqslant^{\text{bool}} \mathcal{N}(-)$. The graphs, however, are distinguishable by the boolean query expressed by $R^2 \circ R^{-1} \circ R^2$. □

As detailed below, Propositions 5.4, 5.6, 5.8 and 6.8 are now subsequently used to show that for every pair F_1 and F_2 of sets of nonbasic features for which $F_1 \not\subseteq \overline{F_2}$ and $\widehat{F_1} \not\subseteq \overline{F_2}$, that $\mathcal{N}(F_1) \not\leqslant^{\text{bool}} \mathcal{N}(F_2)$, in the same way as in Sections 6.1 and 6.2.

The remainder of the proof of Proposition 6.7 is again a combinatorial analysis to verify that the above-mentioned propositions cover all relevant cases.

Proof of Proposition 6.7. If $F_1 \in \mathcal{C}[\cap]$ and $F_2 \in \mathcal{C}$, then $\cap \in \overline{F_1}$ and $\cap \notin \overline{F_2}$. Hence Proposition 5.8 directly implies our result.

Conversely, if $F_1 \in \mathcal{C}$ and $F_2 \in \mathcal{C}[\cap]$, then $x \in \{di, \pi, \overline{\pi}, ^{-1}\}$ is present in F_1, but lacking in $\overline{F_2}$. We will now consider every such x.

If $x \in \{di, \pi, \overline{\pi}\}$ then the proof proceeds as the proof of Proposition 5.5.

If $x = ^{-1}$, then $\widehat{F_1} = (F_1 - \{^{-1}\}) \cup \{\pi\}$ since $F_1 \in \mathcal{C}$. Furthermore, by hypothesis, there is a feature x present in $\widehat{F_1}$ which is not present in $\overline{F_2}$. Notice that $x \neq ^{-1}$. If $x \neq \pi$, then there exists a feature in F_1 other than $^{-1}$ which is missing in $\overline{F_2}$, hence the result follows from the previous case. On the other hand, if $x = \pi$, then $F_2 \cap \{di, \pi, \overline{\pi}, ^{-1}\} = \emptyset$. Hence $F_2 \subseteq \{-, \cap\}$, and thus the result follows directly from Proposition 6.8. □

Propositions 5.1, 6.2, 6.3, 6.5 and 6.7, together prove Theorem 6.1.

Hence, the Hasse diagram of $\leqslant^{\text{bool}}$ can be obtained from the subdiagrams for $\mathcal{C}$, and $\mathcal{C}[\cap]$ by simply adding arrows from $\mathcal{N}$ to $\mathcal{N}(\cap), \mathcal{N}(di, \pi)$ to $\mathcal{N}(\cap, di, \pi), \mathcal{N}(\pi)$ to $\mathcal{N}(\cap, \pi), \mathcal{N}(\overline{\pi}, \pi)$ to $\mathcal{N}(\cap, \overline{\pi}, \pi)$ and $\mathcal{N}(di, \overline{\pi}, \pi)$ to $\mathcal{N}(\cap, di, \overline{\pi}, \pi)$. So, all paths between the subdiagrams are induced by these arrows, the 5 equations from the beginning of Section 5, and Proposition 4.2.

7. Further research

There are alternative modalities for expressing boolean queries apart from interpreting the nonemptiness of an expression as the value true and emptiness as the value false. For example, one possibility is to consider a boolean query q expressible if there are two expressions e_1 and e_2 such that $e_1(G) \subseteq e_2(G)$ if, and only if, $q(G)$ is true, for all G. For some of our

languages, such alternative modalities would not make a difference, but it would for others. Looking into these alternative modalities is an interesting topic for further research.

In the present paper, we have been focusing on expressive power, but, of course, it is also interesting to investigate the decidability of satisfiability or containment of expressions. Much is already known. From the undecidability of FO^3, it follows that the most powerful language is undecidable, and the same holds even without converse. From the decidability of ICPDL [17], all languages without set difference have a decidable satisfiability problem, although this is not yet known for satisfiability restricted to finite relations. An interesting question is the decidability of satisfiability or validity of the languages with set difference, but without the diversity relation. Recently, it has been shown that finite satisfiability for the quite weak fragment $\mathcal{N}(-)$ without *id*, formed by the operators union, composition, set difference and nothing else, over a single binary relation, is still undecidable [35].

Another natural question is whether the notion of arrow logic bisimulation, that we use as a tool to prove some nonexpressibility results, can actually be adapted to obtain characterizations of indistinguishability in the various languages, as is the case for modal logic [18]. We have in fact done this for all languages with intersection [15]. A further question then is whether van Benthem-style expressive completeness results [29] can be established.

Finally, there are still other interesting operators on binary relations that can be considered. A good example is residuation [31], a derived operator of the calculus of relations, and interesting to consider separately, as we have done for projection and coprojection. Residuation is interesting from a database perspective because it corresponds to the set containment join [24].

Acknowledgment

We thank the anonymous referees for their constructive feedback. We thank Balder ten Cate and Maarten Marx for helpful information on the question of succinctness of FO^3 compared to the algebra $\mathcal{N}(di, ^{-1}, -)$.

References

[1] R. Angles, P. Barceló, G. Rios, A practical query language for graph dbs, in: L. Bravo, M. Lenzerini (Eds.), Proceedings 7th Alberto Mendelzon International Workshop on Foundations of Data Management, CEUR Workshop Proceedings, vol. 1087, 2013.
[2] S. Abiteboul, P. Buneman, D. Suciu, Data on the Web: From Relations to Semistructured Data and XML, Morgan Kaufman, 2000.
[3] R. Angles, C. Gutierrez, Survey of graph database models, ACM Comput. Surv. 40 (1) (2008). article 1.
[4] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.
[5] P. Barceló, Querying graph databases, in: Proceedings 32st ACM Symposium on Principles of Databases, ACM, 2013, pp. 175–188.
[6] F. Baader, D. Calvanese, D. McGuiness, D. Nardi, P. Patel-Schneider (Eds.), The Description Logic Handbook, Cambridge University Press, 2003.
[7] M. Benedikt, W. Fan, G.M. Kuper, Structural properties of XPath fragments, Theor. Comput. Sci. 336 (1) (2005) 3–31.
[8] C. Bizer, T. Heath, T. Berners-Lee, Linked data—the story so far, Int. J. Semantic Web Inform. Syst. 5 (3) (2009) 1–22.
[9] P. Blackburn, J. van Benthem, F. Wolter (Eds.), Handbook of Modal Logic, Elsevier, 2007.
[10] H.-D. Ebbinghaus, J. Flum, Finite Model Theory, second ed., Springer, 1999.
[11] K. Etessami, M.Y. Vardi, T. Wilke, First-order logic with two variables and unary temporal logic, Inform. Comput. 179 (2) (2002) 279–295.
[12] G.H.L. Fletcher, M. Gyssens, D. Leinders, D. Surinx, J. Van den Bussche, D. Van Gucht, S. Vansummeren, Y. Wu, Relative expressive power of navigational query on graphs using transitive closure, submitted for publications.
[13] G.H.L. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, S. Vansummeren, Y. Wu, Relative expressive power of navigational querying on graphs, in: Proceedings 14th International Conference on Database Theory, 2011.
[14] G.H.L. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, S. Vansummeren, Y. Wu, The impact of transitive closure on the expressiveness of navigational query languages on unlabeled graphs, Ann. Math. Artif. Intell. (2013). Published online, 2 April.
[15] G.H.L. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, S. Vansummeren, Similarity and bisimilarity notions appropriate for characterizing indistinguishability in fragments of the calculus of relations, J. Logic Comput. (2014). Published online, 25 March 2014.
[16] D. Florescu, A.Y. Levy, A.O. Mendelzon, Database techniques for the World-Wide Web: a survey, SIGMOD Rec. 27 (3) (1998) 59–74.
[17] S. Göller, M. Lohrey, C. Lutz, PDL with intersection and converse: satisfiability and infinite-state model checking, J. Symbolic Logic 74 (1) (2009) 279–314.
[18] V. Goranko, M. Otto, Model theory of modal logic, in: Blackburn et al. [9], 2007, chapter 5.
[19] A. Halevy, M. Franklin, D. Maier, Principles of dataspace systems, in: Proceedings 25th ACM Symposium on Principles of Database Systems, 2006, pp. 1–9.
[20] Robin Hirsch, Ian Hodkinson, Relation Algebras by Games, Elsevier, 2002.
[21] D. Harel, D. Kozen, J. Tiuryn, Dynamic Logic, MIT Press, 2000.
[22] L. Libkin, W. Martens, D. Vrgoč, Quering graph databases with XPath, in: Proceedings 16th International Conference on Database Theory, ACM, 2013.
[23] R.D. Maddux, Relation Algebras, Elsevier, 2006.
[24] N. Mamoulis, Efficient processing of joins on set-valued attributes, in: Proceedings ACM SIGMOD International Conference on Management of Data, 2003, pp. 157–168.
[25] M. Marx, Conditional XPath, ACM Trans. Database Syst. 30 (4) (2005) 929–959.
[26] M. Marx, M. de Rijke, Semantic characterizations of navigational XPath, SIGMOD Rec. 34 (2) (2005) 41–46.
[27] M. Marx, Y. Venema, Multi-Dimensional Modal Logic, Springer, 1997.
[28] D. Olteanu, Forward node-selecting queries over trees, ACM Trans. Database Syst. 32 (1) (2007). article 3.
[29] M. Otto, Model theoretic methods for fragments of FO and special classes of (finite) structures, in: J. Esparza, C. Michaux, C. Steinhorn (Eds.), Finite and Algorithmic Model Theory, Lecture Note Series, vol. 379, London Mathematical Society, 2011. chapter 7.
[30] J. Pérez, M. Arenas, C. Gutierrez, nSPARQL: a navigational language for RDF, J. Web Semantics 8 (4) (2010) 255–270.
[31] V. Pratt, Origins of the calculus of binary relations, in: Proceedings 7th Annual IEEE Symposium on Logic in Computer Science, 1992, pp. 248–254.
[32] RDF primer, W3C Recommendation, February 2004.
[33] A. Tarski, On the calculus of relations, J. Symbolic Logic 6 (1941) 73–89.
[34] A. Tarski, S. Givant, A Formalization of Set Theory Without Variables, AMS Colloquium Publications, vol. 41, American Mathematical Society, 1987.
[35] T. Tan, J. Van den Bussche, X. Zhang, Undecidability of satisfiability in the algebra of finite binary relations with union, composition, and difference, arXiv:1406.0349, 2014.

[36] J. van Benthem, Program constructions that are safe for bisimulation, Studia Logica 60 (1998) 311–330.
[37] Y. Wu, D. Van Gucht, M. Gyssens, J. Paredaens, A study of a positive fragment of path queries: expressiveness, normal form and minimization, Comput. J. 54 (7) (2011) 1091–1118.
[38] XML path language (XPath) version 1.0. W3C Recommendation, November 1999.