The power of Tarski's relation algebra on trees[⋆,⋆⋆]

Jelle Hellings[*]

McMaster University, 1280 Main St. W., Information Technology Building, Hamilton, ON L8S 4L7, Canada and Hasselt University, Martelarenlaan 42, 3500, Hasselt, Belgium

Yuqing Wu

Pomona College, 185 E 6th St., Claremont, CA 91711, USA

Marc Gyssens

Hasselt University, Martelarenlaan 42, 3500, Hasselt, Belgium

Dirk Van Gucht

Indiana University, 919 E 10th St., Bloomington, IN 47408, USA

Abstract

Fragments of Tarski's relation algebra form the basis of many versatile graph and tree query languages including the regular path queries, XPath, and SPARQL. Surprisingly, however, a systematic study of the relative expressive power of relation algebra fragments on trees has not yet been undertaken. In this work, we perform such a systematic study. Our approach is to start from a basic fragment which only allows composition and union. We then study how the expressive power of the query language changes if we add diversity, converse, projections, coprojections, intersection, and/or difference, both for path queries and Boolean queries. For path queries on labeled trees, we found that adding intersection and difference yields more expressive power for some fragments, while adding one of the other operators always yields more expressive power. For Boolean queries on labeled trees, we obtain a similar picture for the relative expressive power, except for a few fragments where adding converse or projection yields no more expressive power. Additionally, we also studied querying unlabeled trees, for which we have found several redundancies. One challenging problem remains open, however, for both path and Boolean queries: does adding difference yield more expressive power to fragments containing at least diversity, coprojections,

⋆This is a revised and extended version of the paper 'The power of Tarski's relation algebra on trees' presented at the 10th International Symposium on Foundations of Information and Knowledge Systems, Budapest, Hungary (FoIKS 2018) [1].

⋆⋆This material is based on work supported by the National Science Foundation under Grant No. NSF 1438990.

*Corresponding author

Email address: jhellings@ucdavis.edu (Jelle Hellings)

and intersection?

Keywords: Tree queries, first-order logic, relation algebra, branching, locality

1. Introduction

Trees can be used to model data that has a hierarchical or nested structure including taxonomies, organizational charts, documents, genealogies, file and directory structures, and the hierarchy of objects in object-oriented programming. We refer to Figure 1 for an example of such a hierarchy. It is therefore not surprising that tree data models have been continuously studied since the 1960s [2–4]. Modern query languages for querying tree data have a heavy reliance on navigating the tree structure. Prime examples of this are XPath [5–8] and the various JSON query languages [9]. At its core, this navigation can be captured by fragments of Tarski's relation algebra [10]. Consequently, tree querying based on fragments of the relation algebra has already been studied in great detail (e.g., [11–14]). Unfortunately, these studies only covered some very basic fragments of the relation algebra, and a comprehensive study of all relation algebra fragments has not yet been undertaken.

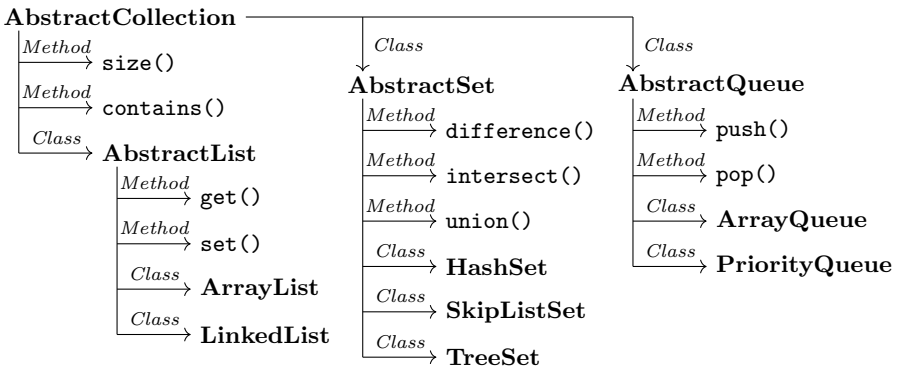

Figure 1: The hierarchical structure of an object-oriented design of a container library. In this design, the nodes represent the individual classes and methods defined in the container library, while the edges represent *Method Of* (Method) and *Class Inheritance* (Class) relationships.

In this work, we carry out such a comprehensive study by investigating the relative expressive power of fragments of the relation algebra with respect to both path queries and Boolean queries. Concretely, the basic relation algebra fragment $\mathcal{N}()$ we start from only allows the constants empty-set and identity ($\emptyset$ and id), edge labels, and the operators composition and union ($\circ$ and $\cup$). This fragment allows for basic querying based on navigating alongside the parent-child axis and corresponds with the first-order fragment of the regular path queries

(RPQs) [15]. As an example, consider the queries

$$q_1 = \mathit{Class} \circ \mathit{Class}$$
$$q_2 = \mathit{Method} \cup (\mathit{Class} \circ \mathit{Method}).$$

The query q_1 yields pairs of classes (m, n) such that n is a subclass of c while c is a subclass of m, and the query q_2 yields class-method pairs (m, n) such that n is a method of m or of a subclass of m.

We study how the expressive power changes if we add the remaining relation algebra constants and operators. This includes adding converse ($^{-1}$) which enables navigation alongside the child-parent axis, yielding the first-order fragment of the 2RPQs [16]. As an example, consider the query

$$q_3 = [\mathit{Class}]^{-1} \circ \mathit{Method},$$

which yields class-method pairs (m, n) such that class m inherits method n from its superclass. We also add projections (π_1 and π_2) which enable simultaneous navigation alongside several branches in the tree, yielding the first-order fragment of the nested RPQs [17]. We can use the projections to *filter and select* nodes of interest. E.g., consider the queries

$$q_4 = \pi_1[\mathit{Class}] \circ \mathit{Method}$$
$$q_5 = \pi_1[\mathit{Method}] \circ \pi_2[\mathit{Class}].$$

The query q_4 yields class-method pairs (m, n) such that n is a method of m, but *only* for classes m that have subclasses. The query q_5 yields pairs (m, m) such that m is a class that defines a method and that is a subclass.

As it turns out, the first-order fragments of the RPQs, 2RPQs, and nested RPQs are rather weak on trees. To increase their expressive power, we consider adding diversity (di) and intersection ($\cap$). The diversity constant evaluates to all pairs of distinct nodes and combined with intersection this constant can be used to, e.g., construct all pairs of distinct siblings. This enables branching and counting queries, even on unlabeled structures. Consider, for example, the queries

$$q_6 = ([\mathit{Class}]^{-1} \circ \mathit{Class}) \cap \mathrm{di}$$
$$q_7 = \pi_1[\mathit{Class} \circ q_6].$$

The query $([\mathit{Class}]^{-1} \circ \mathit{Class})$ yields pairs of classes (m, n) such that m and n are subclasses of some class c. Hence, the query q_6 yields pairs of classes (m, n) such that m and n are *distinct* subclasses of some class c. Finally, the query q_7 yields pairs of classes (c, c) such that c has at-least *two distinct* subclasses. In a similar fashion, we can query for classes that have at-least *three distinct* subclasses:

$$q_8 = \pi_1[\mathit{Class} \circ ((q_6 \circ q_6) \cap \mathrm{di})].$$

Finally, we study adding negation in the form of coprojections ($\overline{\pi}_1$ and $\overline{\pi}_2$) and difference ($-$), which can be used to filter and select nodes by putting restrictions on *navigation* or on the *tree structure*. E.g., consider the queries

$$q_9 = \overline{\pi}_2[\mathit{Class} \cup \mathit{Method}]$$
$$q_{10} = \mathrm{id} - \pi_2[\mathit{Class} \cup \mathit{Method}]$$
$$q_{11} = q_2 \circ \overline{\pi}_1[(\mathrm{di} \cup \mathrm{id}) \circ q_1]$$

Both queries q_9 and q_{10} yield pairs (n, n) such that n cannot be reached by following either the *Class Inheritance* or the *Method Of* relationships. Hence, both queries return the root of the tree. Finally, the query q_{11} yields the result of q_2 whenever query q_1 has an empty result and yields $\emptyset$ otherwise. Hence, in trees in which no class is a subclass of a subclass, q_{11} returns class-method pairs (m, n) such that n is a method of m or of a sublcass of m.

Unfortunately, the relative simplicity of the tree data model turns out to be a curse rather than a blessing: compared to the graph data model [10, 18–20], this simplicity makes it much more difficult to establish separation results using strong brute-force methods. Indeed, one can show that many separation results that can be proven via brute-force results on graphs, cannot be proven via brute-force results on trees. Consequently, the study on trees forces us to search for deeper methods to reach our goals. Therefore, we believe that our study not only gives insight in the expressive power of the relation algebra and its fragments on trees, but also contributes to a better understanding of the fundamental differences between graph data models and tree data models.

The main contribution presented in this paper is the introduction of several properties that can be used to categorize relation algebra fragments according to their expressive power, even with respect to Boolean queries. This in turn yields four major types of separation results on trees. We have summarized these major types in Figure 3. Next, we explore each of these major types of separation results:

Recognizing branches and siblings. The language $\mathcal{N}()$ can only query trees by navigating alongside a single path from ancestor to descendant. Consequently, no query in $\mathcal{N}()$ can distinguish between chains (trees that consist of a single branch from a root to a single leaf) and trees. Other query languages support recognizing branching up to a certain degree, and we can classify these languages accordingly. E.g., queries q_6 and q_7 illustrate how to use $\cap$ and di to recognize branching within trees.

To better characterize the ability to recognize branching, we introduce a notion called *k-subtree reduction* in Section 3. Languages that are closed under k-subtree-reduction steps allow the removal of a child of a node that is structurally equivalent to at least k other children of that node without changing the outcome of Boolean queries. First, we show that the query languages $\mathcal{N}(\mathcal{F})$ (the languages to which the operators in $\mathcal{F}$ are added), $\mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}, \cap\}$, are all 1-*subtree-reducible* and, consequently, can only recognize siblings if they are not structurally equivalent. In very limited circumstances, di can be used to recognize branching.

E.g., the query

$$q_{12} = \mathit{Class} \circ \mathrm{di} \circ \mathrm{di} \circ \mathit{Class}$$

will have an empty result on a two-node tree that contains a single *Class*-labeled edge (Figure 2, *left*), but will not have an empty result if we add another outgoing *Class*-labeled edge to the root (Figure 2, *right*). Hence, any language $\mathcal{N}(\mathcal{F})$ with $\mathrm{di} \in \mathcal{F}$ is *not* 1-subtree reducible and can express queries not expressible in $\mathcal{N}(^{-1}, \pi, \overline{\pi}, \cap)$.

Figure 2: Two trees that can be distinguished by counting the number of children the root has.

Next, we show that query languages $\mathcal{N}(\mathcal{F})$ with $\mathrm{di} \in \mathcal{F}$, but without $\cap, - \notin \mathcal{F}$ are 2-*subtree-reducible*. Finally the full relation algebra is 3-*subtree-reducible*, and, as illustrated by example queries q_6 and q_7, the query languages $\mathcal{N}(\mathcal{F})$ with either $\{^{-1}, -\} \subseteq \mathcal{F}$ or $\{\mathrm{di}, \cap\} \subseteq \mathcal{F}$ can already distinguish between nodes that have one, two, or at least three structurally equivalent children.

Local queries versus non-local queries. Queries in $\mathcal{N}(\mathcal{F})$ with $\mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}, \cap, -\}$ yield node pairs (m, n) such that one can navigate between m and n by traversing a number of edges, with the number depending only on the length of the query. Hence, we call these query languages *local*. Diversity (di) is intrinsically *non-local*. From this observation, it follows that languages with diversity are not path-equivalent to local query languages. This can be strengthened towards Boolean inequivalence, as diversity can, in many cases, be used to express non-local properties on which trees and chains can be distinguished. E.g., the query

$$q_{13} = \mathit{Class} \circ \mathit{Method} \circ \mathrm{di} \circ \mathit{Class} \circ \mathit{Class}$$

will yield a result whenever there is a subclass in the tree with a method and there is a subclass in the three with a subclass.

We do so in Section 4 by exploiting the fact that many properties on which trees and chains can be distinguished are non-local and rely on a limited form of counting. A simple example of this are chain queries of the form "are there k edges in the chain labeled with edge label ℓ".

Downward queries versus non-downward queries. Queries in $\mathcal{N}(\mathcal{F})$ with $\mathcal{F} \subseteq \{\pi, \overline{\pi}, \cap, -\}$ yield node pairs (m, n) such that one can navigate from m to n by traversing along a sequence of parent-child axes, e.g., the queries q_1, q_2, q_4, q_5, q_9, and q_{10}. We call these query languages *downward* [12, 13]. We observe that these downward query languages are all 1-subtree-reducible, which puts an upper bound on their expressive power. Queries that use diversity (di) and converse ($^{-1}$) are *non-downward* in nature, e.g., the queries q_3, q_6, q_7, q_8, q_{11},

q_{12}, and q_{13}. Based on this observation, it follows that languages with diversity or converse are not path-equivalent to downward query languages. For Boolean queries, this is not always the case, as we identify several cases in which adding converse to downward languages does not add expressive power.

Monotonicity. A query language is *monotone* if, for every query q, every graph $\mathcal{G}$, and every graph $\mathcal{G}'$ obtained by adding nodes and edges to $\mathcal{G}$, the result of evaluating query q on $\mathcal{G}$ is contained in the result of evaluating q on $\mathcal{G}'$. Take, e.g., the query q_1 that yields pairs of classes (m, n) such that n is a subclass of a subclass of m. Any existing subclass-of-subclass pairs remain valid whenever we add new information to the tree, making this query monotone. The example query q_9, which returns the root class in the tree, is not monotone, however, as the addition of a parent class of the current root will invalidate the current root.

One the one hand, one can show that the query language $\mathcal{N}(\mathrm{di}, ^{-1}, \pi, \cap)$ is monotone [12, 13]. On the other hand, the query languages $\mathcal{N}(\mathcal{F})$ with $\overline{\pi} \in \mathcal{F}$ (or with sufficient operators in $\mathcal{F}$ to express $\overline{\pi}$) are *non-monotone*. For example, we only need coprojections to construct a Boolean query that puts an upper bound on the length of a chain. E.g., the query

$$q_{14} = \overline{\pi}_2[\mathit{Class}] \circ \mathit{Class} \circ \mathit{Class} \circ \mathit{Class} \circ \overline{\pi}_1[\mathit{Class}]$$

will return a non-empty result if and only if there is a path of length three from the root class to a class without subclasses. Such queries are not monotone and consequently not expressible in $\mathcal{N}(\mathrm{di}, ^{-1}, \pi, \cap)$.

In Figure 3, we visualize the above categorization, which yields an initial classification of the expressive power of the query languages we study on trees. It does not provide all details, however, which we will start to unravel in this paper, mainly in Sections 3 and 4. For an index on how specific results are proven, we refer the reader to Figure 18.

The remaining separation results are obtained through brute-force methods, and are presented in Section 5. Besides separation results, we also establish collapse results in this paper. In Section 6, we obtain these by introducing a notion called condition tree queries for the local relation algebra fragments. They prove to be a powerful tool to show that intersection never adds expressive power beyond the ability to express projections. We also use this tool to establish limitations on the expressive power of projections in Boolean queries.

To further strengthen our results and improve our understanding of both graph querying using Tarski's relation algebra and the tree data model, we have proven each separation result on the simplest data model for which the separation hold (e.g., unlabeled trees or labeled chains). Additionally, in Section 7, we augmented the relation algebra fragments we study with the Kleene-star operator, which is frequently added to practical graph query languages based on the relation algebra, such as the RPQs, 2RPQs, and nested RPQs [15, 16, 21, 22]. For the Kleene-star, we have identified all cases in which adding this non-first-order-definable operator adds expressive power.

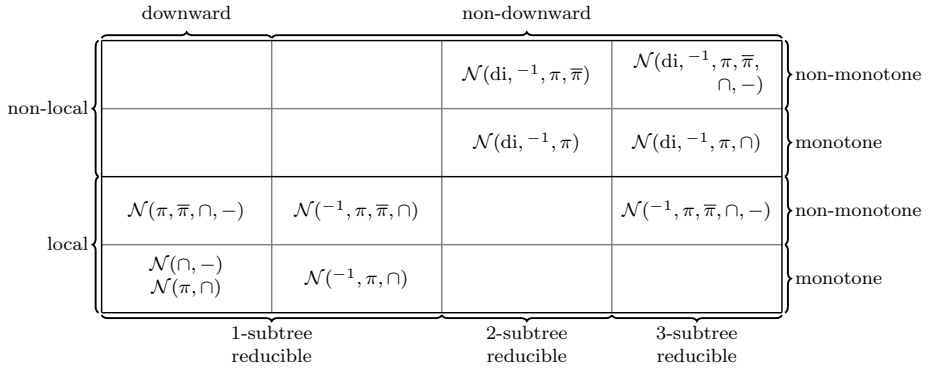

Figure 3: Initial classification of the relative expressive power of fragments of the relation algebra with respect to path queries on labeled trees. In each box, the largest fragment(s) that satisfy the classification of that particular box are included. The more to the right and to the top a certain box is situated, the stronger the expressiveness of the corresponding fragment(s) become.

What remains open is whether adding difference to the fragments containing diversity, coprojections (and hence also projections), and intersection yields a collapse or separation, even in the presence of converse. We argue that this a very challenging open case, and we consider identifying it as the last major contribution of this paper. We discuss this open case in Section 9, in which we also discuss other directions for future research.

This is a revised and extended version of Hellings et al. [1]. Not only did we add full proofs, we also added a complete study of *unlabeled trees*, for which we find several interesting collapses not present in the labeled case. Additionally, we added a study of the transitive closure operator, for which a collapse result is obtained using subtree-reductions. Finally, where possible, our separation results have been proven on chains, strengthening previous results.

2. Preliminaries[1]

Before we introduce query languages to help express indirect relationships, we formalize the graph data model used in this work. We use an *edge-labeled graph data model*:[2]

Definition 2.1. A *graph* is a triple $\mathcal{G} = (\mathcal{V}, \Sigma, \mathbf{E})$, with $\mathcal{V}$ a finite set of nodes, Σ a finite set of edge labels, and $\mathbf{E} : \Sigma \to 2^{\mathcal{V} \times \mathcal{V}}$ a function mapping edge labels

[1] Our formalization of graphs, the relation algebra, and equivalence notions is adapted from concepts used by Fletcher et al. [18, 19].

[2] Various node-labeled graph data models are frequently used in the setting of XML data (see, e.g., [11, 14]). The results presented in this work can easily be adapted to these node-labeled graph data models.

to edge relations. A graph is *unlabeled* if $|\Sigma| = 1$. We use $\mathcal{E}$ to refer to the union of all edge relations, and, when the graph is unlabeled, to the single edge relation. It is said that edge $(m, n) \in \mathcal{E}$ is an *outgoing edge* of m and an *incoming edge* of n. A *tree* $\mathcal{T} = (\mathcal{V}, \Sigma, \mathbf{E})$ is a connected acyclic graph in which exactly one node, the *root*, has no incoming edges, and all other nodes have exactly one incoming edge. If (m, n) is an edge in $\mathcal{T}$, then node m is the *parent* of node n and node n is a *child* of node m. A *chain* is a tree in which all nodes have at most one child.

A *path* of length k is a sequence $n_1\, \ell_1\, n_2 \cdots \ell_k\, n_{k+1}$ with $n_1, \ldots, n_{k+1} \in \mathcal{V}$, $\ell_1, \ldots, \ell_k \in \Sigma$, and, for all $1 \leq i \leq k$, $(n_i, n_{i+1}) \in \mathbf{E}(\ell_j)$. Given a tree, and disregarding the direction of its edges, the *distance* between two nodes is the length of the unique shortest path between them. We write $\|m \leftrightarrow n\|_{\mathcal{T}}$ to denote the distance between nodes m and n in tree $\mathcal{T}$. In general, the unique shortest path between m and n consists of a sequence of upward child-to-parent edges followed by a sequence of downward parent-to-child edges. If one of the nodes is an ancestor of the other, then the unique shortest path between m and n is directed. We define the *directed distance*, denoted by $\|m \to n\|_{\mathcal{T}}$, as $\|m \to n\|_{\mathcal{T}} = \|m \leftrightarrow n\|_{\mathcal{T}}$ if m is an ancestor of n and as $\|m \to n\|_{\mathcal{T}} = -\|m \leftrightarrow n\|_{\mathcal{T}}$ if n is an ancestor of m. Finally, the *depth* of tree $\mathcal{T}$, denoted by $\mathrm{depth}(\mathcal{T})$, is the maximum distance of the root node to any leaf node.

In our setting, a query q maps a graph to a set of node tuples. We write $[\![q]\!]_{\mathcal{G}}$ to denote the *evaluation* of q on graph $\mathcal{G}$. We can interpret a query q literally as a *path query*, or, alternatively, as a *Boolean query*, in which case `true` stands for $[\![q]\!]_{\mathcal{T}} \neq \emptyset$. For simplicity, we assume that queries always yield binary relations (sets of node-pairs, $[\![q]\!]_{\mathcal{G}} \subseteq \mathcal{V} \times \mathcal{V}$). From this perspective, queries that in spirit return nodes will return pairs of identical nodes. Therefore, we refer to queries expressed by relation algebra expressions whose output is a subset of $\{(n, n) \mid n \in \mathcal{V}\}$ as *node expressions*.

In this paper, we limit our study to queries on trees and chains.

Definition 2.2. The *relation algebra* is defined by the grammar

$$e := \emptyset \mid \mathrm{id} \mid \mathrm{di} \mid \ell \mid [e]^{-1} \mid \pi_j[e] \mid \overline{\pi}_j[e] \mid e \circ e \mid e \cup e \mid e \cap e \mid e - e,$$

in which $\ell \in \Sigma$ and $j \in \{1, 2\}$. Let $\mathcal{T} = (\mathcal{V}, \Sigma, \mathbf{E})$ be a tree and let e be an expression. The semantics of evaluation is defined as follows:

$$\begin{aligned}
[\![\emptyset]\!]_{\mathcal{T}} &= \emptyset; \\
[\![\mathrm{id}]\!]_{\mathcal{T}} &= \{(m, m) \mid m \in \mathcal{V}\}; \\
[\![\mathrm{di}]\!]_{\mathcal{T}} &= \{(m, n) \mid m, n \in \mathcal{V} \wedge m \neq n\}; \\
[\![\ell]\!]_{\mathcal{T}} &= \mathbf{E}(\ell); \\
[\![[e]^{-1}]\!]_{\mathcal{T}} &= \{(n, m) \mid (m, n) \in [\![e]\!]_{\mathcal{T}}\}; \\
[\![\pi_1[e]]\!]_{\mathcal{T}} &= \{(m, m) \mid (m, n) \in [\![e]\!]_{\mathcal{T}}\}; \\
[\![\pi_2[e]]\!]_{\mathcal{T}} &= \{(n, n) \mid (m, n) \in [\![e]\!]_{\mathcal{T}}\}; \\
[\![\overline{\pi}_j[e]]\!]_{\mathcal{T}} &= [\![\mathrm{id}]\!]_{\mathcal{T}} - [\![\pi_j[e]]\!]_{\mathcal{T}};
\end{aligned}$$

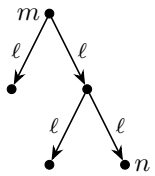

Figure 4: A labeled tree that matches $\pi_1[\ell] \circ \ell \circ \pi_1[\ell] \circ \ell$ exactly.

$$\begin{aligned}
[\![e_1 \circ e_2]\!]_{\mathcal{T}} &= \{(m,n) \mid \exists z\ (m,z) \in [\![e_1]\!]_{\mathcal{T}} \wedge (z,n) \in [\![e_2]\!]_{\mathcal{T}}\};\\
[\![e_1 \cup e_2]\!]_{\mathcal{T}} &= [\![e_1]\!]_{\mathcal{T}} \cup [\![e_2]\!]_{\mathcal{T}};\\
[\![e_1 \cap e_2]\!]_{\mathcal{T}} &= [\![e_1]\!]_{\mathcal{T}} \cap [\![e_2]\!]_{\mathcal{T}};\\
[\![e_1 - e_2]\!]_{\mathcal{T}} &= [\![e_1]\!]_{\mathcal{T}} - [\![e_2]\!]_{\mathcal{T}}.
\end{aligned}$$

In line with Definition 2.1, we write $\mathcal{E}$ to denote the relation algebra expression obtained by taking the union of all edge labels in Σ.

For structural induction purposes, we defined the length of an expression recursively with the base cases $\text{length}(e) = 1$ whenever e is of the form $\emptyset$, id, di, or $\ell \in \Sigma$, the recursive cases $\text{length}(e) = \text{length}(e') + 1$ if e is of the form $[e']^{-1}$, $\pi_j[e']$, or $\overline{\pi}_j[e']$, $j \in \{1,2\}$, and, finally, the recursive cases $\text{length}(e) = \max(\text{length}(e_1), \text{length}(e_2)) + 1$ if e is of the form $e_1 \circ e_2$, $e_1 \cup e_2$, $e_1 \cap e_2$, or $e_1 - e_2$.

One can push converse operators $(^{-1})$ downwards in expressions. Hence, to simplify proofs in this work, we frequently assume that the usage of converse is always restricted to edge labels, i.e., $[\ell]^{-1}$, $\ell \in \Sigma$. Likewise, one can pull union operators $(\cup)$ upwards in expressions. This allows us to assume that expressions are written as unions of $\cup$-free expressions.

Example 2.3. Consider the labeled tree in Figure 4. The expression $e = \pi_1[\ell] \circ \ell \circ \pi_1[\ell] \circ \ell$ matches this tree structure, and will return the node pair (m,n). The expressions $([\ell]^{-1} \circ \ell) \cap \text{di}$ and $([\ell]^{-1} \circ \ell) - \text{id}$ both return pairs of siblings in the tree.

Let e be an expression and k an integer. We write

$$e^k = \begin{cases} e \circ e^{k-1} & \text{if } k > 0;\\ \text{id} & \text{if } k = 0;\\ [e]^{-1} \circ e^{k+1} & \text{if } k < 0. \end{cases}$$

Hence, e^k, $k > 0$, denotes the k-fold composition of e and e^{-k} denotes its converse. We use $[\mathcal{E}]^+$ to denote the *descendant-axis* defined by $[\mathcal{E}]^+ = \bigcup_{k>0} \mathcal{E}^k$, and $[[\mathcal{E}]^{-1}]^+$ to denote the *ancestor-axis* defined by $[[\mathcal{E}]^{-1}]^+ = \bigcup_{k>0} \mathcal{E}^{-k}$.

We write $\mathcal{F} \subseteq \{\text{di}, ^{-1}, \pi, \overline{\pi}, \cap, -\}$ to denote a set of operators in which π represents both π_1 and π_2 and, likewise, $\overline{\pi}$ represents both $\overline{\pi}_1$ and $\overline{\pi}_2$. By $\mathcal{N}(\mathcal{F})$ we denote the fragment of the relation algebra that only allows $\emptyset$, id, $\ell \in \Sigma$, $\circ$, $\cup$, and all operators in $\mathcal{F}$. In this work, we primarily study the expressive

power of the relation algebra and its fragments. To do so, we introduce the appropriate equivalence notions. We consider two such notions: path equivalence and Boolean equivalence.

Definition 2.4. Let q_1 and q_2 be queries and let $\mathbb{G}$ be a class of graphs (e.g., labeled graphs, unlabeled graphs, labeled trees, unlabeled trees, labeled chains, or unlabeled chains). We say that q_1 and q_2 are *path-equivalent* on $\mathbb{G}$, denoted by $q_1 \equiv_{\text{path},\mathbb{G}} q_2$, if, for every graph $\mathcal{G} \in \mathbb{G}$, $[\![q_1]\!]_{\mathcal{G}} = [\![q_2]\!]_{\mathcal{G}}$ and are *Boolean-equivalent* on $\mathbb{G}$, denoted by $q_1 \equiv_{\text{bool},\mathbb{G}} q_2$, if, for every graph $\mathcal{G} \in \mathbb{G}$, $[\![q_1]\!]_{\mathcal{G}} = \emptyset$ if and only if $[\![q_2]\!]_{\mathcal{G}} = \emptyset$.

Whenever the class of graphs $\mathbb{G}$ is clear from the context, we simply write $q_1 \equiv_{\text{path}} q_2$ and $q_1 \equiv_{\text{bool}} q_2$ to denote path-equivalence and Boolean-equivalence on $\mathbb{G}$, respectively.

Expressions that are path-equivalent are also Boolean-equivalent, but the reverse is generally not true:

Example 2.5. The equivalence $e_1 \cap e_2 \equiv_{\text{path}} e_1 - (e_1 - e_2)$ on graphs is well-known. Hence, also $e_1 \cap e_2 \equiv_{\text{bool}} e_1 - (e_1 - e_2)$. On graphs, we also have $\pi_1[\ell] \equiv_{\text{bool}} \ell \equiv_{\text{bool}} \pi_2[\ell]$, but even on chains we have $\pi_1[\ell] \not\equiv_{\text{path}} \ell$ and $\ell \not\equiv_{\text{path}} \pi_2[\ell]$. Finally, let $e = \pi_1[\ell] \circ \ell \circ \pi_1[\ell] \circ \ell$ be the expression e from Example 2.3. On trees, we have $e \equiv_{\text{path}} \ell \circ [\ell]^{-1} \circ \ell \circ \ell \circ [\ell]^{-1} \circ \ell$. This is no longer true on general graphs, however.

The equivalence notions introduced extend naturally to subsumption and equivalence notions between classes of expressions.

Definition 2.6. Let $sem \in \{\text{path}, \text{bool}\}$ and let $\mathbb{G}$ be a class of graphs (e.g., labeled graphs, unlabeled graphs, labeled trees, unlabeled trees, labeled chains, or unlabeled chains). We say that the class of queries $\mathcal{L}_1$ is *sem-subsumed* by the class of queries $\mathcal{L}_2$ on $\mathbb{G}$, denoted by $\mathcal{L}_1 \preceq_{sem,\mathbb{G}} \mathcal{L}_2$, if every query in $\mathcal{L}_1$ is *sem*-equivalent on $\mathbb{G}$ to a query in $\mathcal{L}_2$. Hence,

$$\mathcal{L}_1 \preceq_{sem,\mathbb{G}} \mathcal{L}_2 \text{ if and only if } \forall_{q_1 \text{ in } \mathcal{L}_1} \exists_{q_2 \text{ in } \mathcal{L}_2} q_1 \equiv_{sem,\mathbb{G}} q_2.$$

We say that the classes of queries $\mathcal{L}_1$ and $\mathcal{L}_2$ are *sem-equivalent* on $\mathbb{G}$, denoted by $\mathcal{L}_1 \equiv_{sem,\mathbb{G}} \mathcal{L}_2$, if $\mathcal{L}_1 \preceq_{sem,\mathbb{G}} \mathcal{L}_2$ and $\mathcal{L}_2 \preceq_{sem,\mathbb{G}} \mathcal{L}_1$.

Whenever the class of graphs $\mathbb{G}$ is clear from the context, we simply write $\mathcal{L}_1 \preceq_{sem} \mathcal{L}_2$ and $\mathcal{L}_1 \equiv_{sem} \mathcal{L}_2$ to denote *sem*-subsumption and *sem*-equivalence on $\mathbb{G}$, respectively.

If expressions in one language are always part of another language, then we obviously have path-subsumption:

Example 2.7. On general graphs, we have $\mathcal{N}(^{-1}, \pi) \preceq_{\text{path}} \mathcal{N}(^{-1}, \pi, \cap)$ as every expression in $\mathcal{N}(^{-1}, \pi)$ is an expression in $\mathcal{N}(^{-1}, \pi, \cap)$. Now consider a language $\mathcal{N}(\mathcal{F})$ with $- \in \mathcal{F}$. As illustrated in Example 2.5, we can express the operator $\cap$ using only $-$. Hence, we have $\mathcal{N}(\mathcal{F}) \equiv_{\text{path}} \mathcal{N}(\mathcal{F} - \{\cap\})$ as well.

Examples 2.5 and 2.7 illustrate that several fragments of the relation algebra can express exactly the same set of queries: if, for example, intersection is missing from a fragment that has difference, then difference can be used instead to express intersection. We observe that the following rewrite rules can be used to express operators using other operators:

$$\begin{aligned}
\pi_1[e] &= \pi_2[[e]^{-1}] = \overline{\pi}_j[\overline{\pi}_1[e]] = (e \circ [e]^{-1}) \cap \mathrm{id} = (e \circ (\mathrm{id} \cup \mathrm{di})) \cap \mathrm{id};\\
\pi_2[e] &= \pi_1[[e]^{-1}] = \overline{\pi}_j[\overline{\pi}_2[e]] = ([e]^{-1} \circ e) \cap \mathrm{id} = ((\mathrm{id} \cup \mathrm{di}) \circ e) \cap \mathrm{id};\\
\overline{\pi}_1[e] &= \overline{\pi}_2[[e]^{-1}] = \mathrm{id} - \pi_1[e];\\
\overline{\pi}_2[e] &= \overline{\pi}_1[[e]^{-1}] = \mathrm{id} - \pi_2[e];\\
e_1 \cap e_2 &= e_1 - (e_1 - e_2),
\end{aligned}$$

with $j \in \{1, 2\}$. Let $\mathcal{F} \subseteq \{\mathrm{di}, ^{-1}, \pi, \overline{\pi}, \cap, -\}$. By $\underline{\mathcal{F}}$ we denote the set of operators obtained from $\mathcal{F}$ by adding operators to $\mathcal{F}$ that can be expressed using the operators already in $\mathcal{F}$ using the above rules.

As an alternative to the relation algebra, we can also consider arbitrary first-order logic formulae to specify graph queries. Let $\mathcal{G} = (\mathcal{V}, \Sigma, \mathbf{E})$ be a graph with $\Sigma = \{\ell_1, \ldots, \ell_{|\Sigma|}\}$. We write FO[$k$] to denote the first-order logic over the structure $(\mathcal{V}; \ell_1, \ldots, \ell_{|\Sigma|})$ in which ℓ_i, $1 \leq i \leq |\Sigma|$, represents the edge relation $\mathbf{E}(\ell_i)$, and in which variables are restricted to a set of at most k variables. By $\mathrm{L}^k_{\infty\omega}$, we denote the standard infinitary first-order logic extension of FO[k] that allows conjunctions and disjunctions over arbitrary sets [23–25].

3. Detecting branches and subtree-reductions

The obvious way to detect whether or not a tree is a chain is by checking whether some node has several children. This is particularly easy when these children are connected to their parent using distinct edge labels. Hellings et al. [13] already showed that only the most basic language is not capable of detecting labeled branching. In contrast, most other languages are able to detect labeled branching:

Example 3.1 (Hellings et al. [13, Proposition 3.6]). Consider the expressions $e_1 = [\ell_1]^{-1} \circ \ell_2$ and $e_2 = \pi_1[\ell_1] \circ \pi_1[\ell_2]$. For any tree $\mathcal{T}$, we have $[\![e_1]\!]_{\mathcal{T}} \neq \emptyset$ and $[\![e_2]\!]_{\mathcal{T}} \neq \emptyset$ if and only if $\mathcal{T}$ has a node with at least two children, one reachable via an edge labeled ℓ_1 and another via an edge labeled ℓ_2.

Detecting branches in the situation above, where a single node has several structurally distinct branches, is relatively simple. This changes when branches are structurally identical. As a first step towards our goal of identifying language fragments in which structurally identical branches can be distinguished, we take advantage of the simple structure of trees to exhibit limitations on the expressive power of relation algebra fragments. Thereto, we introduce *subtree-reductions*.

Let $k > 0$. A *k-subtree-reduction step* on tree $\mathcal{T} = (\mathcal{V}, \Sigma, \mathbf{E})$ consists of first finding different nodes $m, n_1, \ldots, n_{k+1} \in \mathcal{V}$ and an edge label $\ell \in \Sigma$ such that the subtrees rooted at $n_1, \ldots, n_{k+1}$ are isomorphic and $(m, n_1), \ldots, (m, n_{k+1}) \in$

Figure 5: The tree $\mathcal{T}_1$ representing a chain (*left*), tree $\mathcal{T}_2$ (*middle*), and tree $\mathcal{T}_3$ (*right*), which can be distinguished by counting the number of children of the root in these trees.

$\mathbf{E}(\ell)$, and then picking a node n_i, $1 \leq i \leq k+1$, and removing the subtree rooted at n_i.

Definition 3.2. We say that a tree is *k-subtree-reducible* if we can apply a k-subtree-reduction step.[3]

Example 3.3. Consider the unlabeled trees $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ in Figure 5. The tree $\mathcal{T}_1$ can be obtained by a 1-subtree-reduction step on $\mathcal{T}_2$ and $\mathcal{T}_2$ can be obtained by a 2-subtree-reduction step on $\mathcal{T}_3$. Consequently, $\mathcal{T}_1$ can also be obtained by two 1-subtree-reduction steps on $\mathcal{T}_3$. Hence, $\mathcal{T}_2$ is 1-subtree-reducible and $\mathcal{T}_3$ is 1-subtree-reducible and 2-subtree-reducible.

We now exhibit conditions under which the result of a relation algebra expression is invariant under subtree-reductions at the Boolean level. First, we conclude the following:

Lemma 3.4. *Let $\mathcal{T}$ be a tree and let $\mathcal{T}'$ be obtained from $\mathcal{T}$ by a k-subtree-reduction step. For every query q in $\mathrm{L}^k_{\infty\omega}$, we have $[\![q]\!]_{\mathcal{T}} \neq \emptyset$ if and only if $[\![q]\!]_{\mathcal{T}'} \neq \emptyset$.*

Proof. To prove this lemma, we only have to show that the duplicator has a winning strategy in any k-pebble game played on $\mathcal{T}$ and $\mathcal{T}'$ [24, Theorem 3.7]. Let $\mathcal{T} = (\mathcal{V}, \Sigma, \mathbf{E})$ and consider the k-subtree-reduction step that resulted in $\mathcal{T}' = (\mathcal{V}', \Sigma, \mathbf{E}')$. This k-subtree-reduction step operated on node $m \in \mathcal{V}$ with children $n_1, \ldots, n_{k+1} \in \mathcal{V}$ such that the subtrees rooted at $n_1, \ldots, n_{k+1}$ are isomorphic and $(m, n_1), \ldots, (m, n_{k+1}) \in \mathbf{E}(\ell)$ for some edge label $\ell \in \Sigma$. Without loss of generality, we may assume that the subtree rooted at n_{k+1} was removed to obtain $\mathcal{T}'$. Hence, if $\mathcal{V}|_{n_{k+1}}$ are the nodes in the subtree rooted at n_{k+1}, then $\mathcal{V}' = \mathcal{V} - \mathcal{V}|_{n_{k+1}}$ and, for all $\ell' \in \Sigma$, $\mathbf{E}'(\ell') = \mathbf{E}(\ell') - (\mathcal{V} \times \mathcal{V}|_{n_{k+1}})$.

Now consider any k-pebble game played on $\mathcal{T}$ and $\mathcal{T}'$. Trees $\mathcal{T}$ and $\mathcal{T}'$ only differ in the subtrees isomorphic to the subtree rooted at n_1: $\mathcal{T}$ has at-least $k+1$ such subtrees reachable via an edge label ℓ from node m, whereas $\mathcal{T}'$ has at-least k such subtrees reachable via an edge label ℓ from node m. Whenever the spoiler places a pebble on a node outside the subtrees rooted at $n_1, \ldots, n_{k+1}$ in either $\mathcal{T}$

[3]The 1-subtree-reduction step bears a close relationship to the F+B-index and the F&B-index used for indexing the structure of tree data [26]. As with the F+B-index, the 1-subtree-reduction steps will only merge child nodes when both the forward structure (subtrees rooted at the children) and backward structure (children of a single parent) "behave equivalently" (in the sense that merged nodes allow the same edge-based navigation). The k-subtree-reduction step, $k > 1$, generalizes the principles of the F+B- and F&B-indices by not only taking into account forward and backward structure, but also taking into account how often (up-to-k) the forward structures occur.

or $\mathcal{T}'$, the duplicator can match this play in the other tree structure. For plays within the subtrees rooted at $n_1, \ldots n_{k+1}$ in either $\mathcal{T}$ or $\mathcal{T}'$, we observe that only k pebbles of the spoiler are in play. Hence, at most k subtrees rooted at $n_1, \ldots n_{k+1}$ in $\mathcal{T}$ or $\mathcal{T}'$ have pebbles of the spoiler in them, and the duplicator can always maintain a one-to-one correspondence between the at-most-k subtrees in $\mathcal{T}$ or $\mathcal{T}'$ that have pebbles of the spoiler in them. To respond by any plays of the spoiler within the subtrees rooted at $n_1, \ldots n_{k+1}$ in either $\mathcal{T}$ or $\mathcal{T}'$, the duplicator simply plays in accordance with this one-to-one correspondence between trees and matches this play in the other tree structure. □

It is well known that $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap, -) \preceq_{\mathrm{path}} \mathrm{FO}[3]$ [10, 20]. By a result of Hellings et al. [27, Theorem 6.1], we also have $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi}) \preceq_{\mathrm{bool}} \mathrm{FO}[2]$. Combining this with Lemma 3.4 yields the following:

Proposition 3.5. *Let $\mathcal{F} \subseteq \{\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap, -\}$, let e be an expression in $\mathcal{N}(\mathcal{F})$, let $\mathcal{T}$ be a tree, and let $\mathcal{T}'$ be obtained from $\mathcal{T}$ by a k-subtree-reduction step.*

1. *If $k \geq 3$, then $[\![e]\!]_{\mathcal{T}} \neq \emptyset$ if and only if $[\![e]\!]_{\mathcal{T}'} \neq \emptyset$.*
2. *If $k \geq 2$ and $\cap \notin \underline{\mathcal{F}}$, then $[\![e]\!]_{\mathcal{T}} \neq \emptyset$ if and only if $[\![e]\!]_{\mathcal{T}'} \neq \emptyset$.*

For 1-subtree-reduction steps, we can further strengthen Proposition 3.5:

Proposition 3.6. *Let $\mathcal{F} \subseteq \{{}^{-1}, \pi, \overline{\pi}, \cap\}$, let e be an expression in $\mathcal{N}(\mathcal{F})$, let $\mathcal{T}$ be a tree, and let $\mathcal{T}'$ be obtained from $\mathcal{T}$ by a 1-subtree-reduction step. We have $[\![e]\!]_{\mathcal{T}} \neq \emptyset$ if and only if $[\![e]\!]_{\mathcal{T}'} \neq \emptyset$.*

Proof. Let $\mathcal{T} = (\mathcal{V}, \Sigma, \mathbf{E})$, let $\mathcal{T}' = (\mathcal{V}', \Sigma', \mathbf{E}')$, and let $n_1, n_2 \in \mathcal{V}$ be siblings in $\mathcal{T}$ such that the subtrees rooted at n_1 and n_2 are isomorphic and $\mathcal{T}'$ is obtained from $\mathcal{T}$ by eliminating the subtree rooted at n_2. Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be the sets of nodes in the subtrees of $\mathcal{T}$ rooted at n_1 and n_2, respectively, and let $b : \mathcal{V}_1 \to \mathcal{V}_2$ be the bijection establishing that these subtrees are isomorphic. Let g be the identity on $\mathcal{V} - (\mathcal{V}_1 \cup \mathcal{V}_2)$, and let $f = b \cup b^{-1} \cup g$. Since f is an automorphism of $\mathcal{T}$, we have

(1) if $m, n \in \mathcal{V}$, then $(m, n) \in [\![e]\!]_{\mathcal{T}}$ if and only if $(f(m), f(n)) \in [\![e]\!]_{\mathcal{T}}$.

By induction on the length of e, we next prove that we have, in addition,

(2) if $m \in \mathcal{V}_1, n \in \mathcal{V}_2$ or $m \in \mathcal{V}_2, n \in \mathcal{V}_1$, then $(m, n) \in [\![e]\!]_{\mathcal{T}}$ implies $(f(m), n) \in [\![e]\!]_{\mathcal{T}}$;
(3) if $m, n \in \mathcal{V}'$, then $(m, n) \in [\![e]\!]_{\mathcal{T}}$ if and only if $(m, n) \in [\![e]\!]_{\mathcal{T}'}$.

Observe that $f = f^{-1}$. Hence, Property (2) also yields $(m, n) \in [\![e]\!]_{\mathcal{T}}$ implies $(m, f(n)) \in [\![e]\!]_{\mathcal{T}}$. The base cases are expressions of the form $\emptyset$, id, ℓ, and $[\ell]^{-1}$, with ℓ an edge label, for which it is straightforward to verify that the Properties (2) and (3) hold. Now assume that the Properties hold for all expressions e of length at most i. Let e be an expression of length $i + 1$. We distinguish the following cases:

i. $e = \overline{\pi}_1[e']$, *Property (2).* We have $(m, n) \in [\![e]\!]_{\mathcal{T}}$ only if $m = n$. Hence, the premise of Property (2) is not applicable to e in this case, and we conclude that the Property holds voidly.

ii. $e = \overline{\pi}_1[e']$, *Property (3).* We assume $m, n \in \mathcal{V}'$. If $(m, n) \in [\![e]\!]_{\mathcal{T}}$, then $m = n$ and, for all $z \in \mathcal{V}$, $(m, z) \notin [\![e']\!]_{\mathcal{T}}$. In particular, this is the case for all $z \in \mathcal{V}'$. By applying Property (3) on e', we obtain $(m, z) \notin [\![e']\!]_{\mathcal{T}'}$ for all z. Hence, we conclude $(m, n) \in [\![e]\!]_{\mathcal{T}'}$.
If $(m, n) \in [\![e]\!]_{\mathcal{T}'}$, then $m = n$ and, for all $z' \in \mathcal{V}$, $(m, z') \notin [\![e']\!]_{\mathcal{T}'}$. By applying Property (3) on e', we obtain $(m, z') \notin [\![e']\!]_{\mathcal{T}}$ for all $z' \in \mathcal{V}'$. Let $z'' \notin \mathcal{V}'$. Hence, $z'' \in \mathcal{V}_2$, $f(z'') \in \mathcal{V}'$, and $(m, f(z'')) \notin [\![e']\!]_{\mathcal{T}}$. If $m \in \mathcal{V}_1$, then we apply Property (2) on e' to obtain $(m, z'') \in [\![e']\!]_{\mathcal{T}}$ implies $(m, f(z'')) \in [\![e']\!]_{\mathcal{T}}$. Else, if $m \notin \mathcal{V}_1$, then $f(m) = m$ and we apply Property (1) on e' to obtain $(m, z'') \in [\![e']\!]_{\mathcal{T}}$ implies $(m, f(z'')) \in [\![e']\!]_{\mathcal{T}}$. Hence, in both cases, $(m, z'') \notin [\![e']\!]_{\mathcal{T}}$ must hold. We conclude $(m, n) \in [\![e]\!]_{\mathcal{T}}$.

iii. $e = e_1 \circ e_2$, *Property (2).* We assume $m \in \mathcal{V}_1, n \in \mathcal{V}_2$. The case for $m \in \mathcal{V}_2, n \in \mathcal{V}_1$ is analogous. We have $(m, n) \in [\![e]\!]_{\mathcal{T}}$ if there exists z such that $(m, z) \in [\![e_1]\!]_{\mathcal{T}}$ and $(z, n) \in [\![e_2]\!]_{\mathcal{T}}$. We distinguish three cases.
   (a) $z \in \mathcal{V}_1$. We apply Property (1) on e_1 and Property (2) on e_2 to obtain $(f(m), f(z)) \in [\![e_1]\!]_{\mathcal{T}}$ and $(f(z), n) \in [\![e_2]\!]_{\mathcal{T}}$. Hence, $(f(m), n) \in [\![e]\!]_{\mathcal{T}}$.
   (b) $z \in \mathcal{V}_2$. We apply Property (2) on e_1 to obtain $(f(m), z) \in [\![e_1]\!]_{\mathcal{T}}$. Hence, $(f(m), n) \in [\![e]\!]_{\mathcal{T}}$.
   (c) $z \notin \mathcal{V}_1$ *and* $z \notin \mathcal{V}_2$. We have $f(z) = z$ and we apply Property (1) on e_1 to obtain $(f(m), z) \in [\![e_1]\!]_{\mathcal{T}}$. Hence, $(f(m), n) \in [\![e]\!]_{\mathcal{T}}$.

iv. $e = e_1 \circ e_2$, *Property (3).* We assume $m, n \in \mathcal{V}'$. If $(m, n) \in [\![e]\!]_{\mathcal{T}}$, then there exists $z \in \mathcal{V}$ such that $(m, z) \in [\![e_1]\!]_{\mathcal{T}}$ and $(z, n) \in [\![e_2]\!]_{\mathcal{T}}$. If $z \in \mathcal{V}'$, then we apply Property (3) on e_1 and e_2 to obtain $(m, z) \in [\![e_1]\!]_{\mathcal{T}'}$ and $(z, n) \in [\![e_2]\!]_{\mathcal{T}'}$. Hence, $(m, n) \in [\![e]\!]_{\mathcal{T}'}$. If $z \notin \mathcal{V}'$, then $z \in \mathcal{V}_2$ and $f(z) \in \mathcal{V}'$. If, additionally, $m \in \mathcal{V}_1$, then we apply Property (2) on e_1 to obtain $(m, f(z)) \in [\![e_1]\!]_{\mathcal{T}}$. Else, if $m \notin \mathcal{V}_1$, then $f(m) = m$ and we apply Property (1) on e_1 to obtain $(m, f(z)) \in [\![e_1]\!]_{\mathcal{T}}$. Likewise, we can also obtain $(f(z), n) \in [\![e_2]\!]_{\mathcal{T}}$. We apply Property (3) on e_1 and e_2 to obtain $(m, f(z)) \in [\![e_1]\!]_{\mathcal{T}'}$ and $(f(z), n) \in [\![e_2]\!]_{\mathcal{T}'}$. Hence, $(m, n) \in [\![e]\!]_{\mathcal{T}'}$.
If $(m, n) \in [\![e]\!]_{\mathcal{T}'}$, then there exists $z' \in \mathcal{V}'$ such that $(m, z') \in [\![e_1]\!]_{\mathcal{T}'}$ and $(z', n) \in [\![e_2]\!]_{\mathcal{T}'}$. By applying Property (3) on e_1 and e_2, we obtain $(m, z') \in [\![e_1]\!]_{\mathcal{T}}$, $(z', n) \in [\![e_2]\!]_{\mathcal{T}}$, and we conclude $(m, n) \in [\![e]\!]_{\mathcal{T}}$.

v. $e = e_1 \cup e_2$, *Property (2).* We assume $m \in \mathcal{V}_1, n \in \mathcal{V}_2$. The case for $m \in \mathcal{V}_2, n \in \mathcal{V}_1$ is analogous. We have $(m, n) \in [\![e]\!]_{\mathcal{T}}$ if $(m, n) \in [\![e_1]\!]_{\mathcal{T}}$ or $(m, n) \in [\![e_2]\!]_{\mathcal{T}}$. We apply Property (2) on e_1 and e_2 to obtain $(f(m), n) \in [\![e_1]\!]_{\mathcal{T}}$ or $(f(m), n) \in [\![e_2]\!]_{\mathcal{T}}$. Hence, we conclude $(f(m), n) \in [\![e]\!]_{\mathcal{T}}$.

vi. $e = e_1 \cup e_2$, *Property (3).* We assume $m, n \in \mathcal{V}'$. We have $(m, n) \in [\![e]\!]_{\mathcal{T}}$ if and only if $(m, n) \in [\![e_1]\!]_{\mathcal{T}}$ or $(m, n) \in [\![e_2]\!]_{\mathcal{T}}$. By applying Property (3) on e_1 and e_2, we have $(m, n) \in [\![e_1]\!]_{\mathcal{T}}$ or $(m, n) \in [\![e_2]\!]_{\mathcal{T}}$ if and only if $(m, n) \in [\![e_1]\!]_{\mathcal{T}'}$ or $(m, n) \in [\![e_2]\!]_{\mathcal{T}'}$. Hence, we conclude $(m, n) \in [\![e]\!]_{\mathcal{T}}$ if and only if $(m, n) \in [\![e]\!]_{\mathcal{T}'}$.

The case for $e = e_1 \cap e_2$ is analogous to the case for $e = e_1 \cup e_2$ by replacing "or" by "and". The case for $e = \pi_1[e']$ can be obtained by rewriting $\pi_1[e']$ to $\overline{\pi}_1[\overline{\pi}_1[e']]$. The cases for $e = \pi_2[e']$ and $e = \overline{\pi}_2[e']$ are analogous to $e = \pi_1[e']$ and $e = \overline{\pi}_1[e']$, respectively, completing our proof. □

From the limitations imposed by Proposition 3.5 and Proposition 3.6 on the expressive power of the fragments considered, we deduce the following separation results:

Proposition 3.7. *Let* $\mathcal{F}_1, \mathcal{F}_2 \subseteq \{\mathrm{di}, ^{-1}, \pi, \overline{\pi}, \cap\}$ *with* $\mathrm{di} \notin \mathcal{F}_1$ *and* $\cap \notin \underline{\mathcal{F}_2}$. *Already on unlabeled trees, we have*

1. $\mathcal{N}(\mathrm{di}) \not\leq_{\mathrm{bool}} \mathcal{N}(\mathcal{F}_1)$ *and* $\mathcal{N}(^{-1}, -) \not\leq_{\mathrm{bool}} \mathcal{N}(\mathcal{F}_1)$ *and*
2. $\mathcal{N}(\mathrm{di}, \cap) \not\leq_{\mathrm{bool}} \mathcal{N}(\mathcal{F}_2)$ *and* $\mathcal{N}(^{-1}, -) \not\leq_{\mathrm{bool}} \mathcal{N}(\mathcal{F}_2)$.

Proof. Consider the unlabeled trees $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ in Example 3.3. Since $\mathcal{T}_1$ can be obtained by a 1-subtree-reduction on $\mathcal{T}_2$, we have, by Proposition 3.6, that, for every e in $\mathcal{N}(\mathcal{F}_1)$, $[\![e]\!]_{\mathcal{T}_2} \neq \emptyset \iff [\![e]\!]_{\mathcal{T}_1} \neq \emptyset$. Now consider the expressions $e_1 = \ell \circ \mathrm{di} \circ \mathrm{di} \circ \ell$ in $\mathcal{N}(\mathrm{di})$ and $e_2 = ([\ell]^{-1} \circ \ell) - \mathrm{id}$ in $\mathcal{N}(^{-1}, -)$. We have $[\![e_1]\!]_{\mathcal{T}_2} \neq \emptyset$ and $[\![e_2]\!]_{\mathcal{T}_2} \neq \emptyset$, while $[\![e_1]\!]_{\mathcal{T}_1} = [\![e_2]\!]_{\mathcal{T}_1} = \emptyset$.

Since $\mathcal{T}_2$ can be obtained by a 2-subtree-reduction on $\mathcal{T}_3$, we have, by Proposition 3.5(2), that, for every e in $\mathcal{N}(\mathcal{F}_2)$, $[\![e]\!]_{\mathcal{T}_3} \neq \emptyset \iff [\![e]\!]_{\mathcal{T}_2} \neq \emptyset$. Now consider the expressions $e_3 = (((\mathrm{di} \circ \ell) \cap \mathrm{di}) \circ ((\mathrm{di} \circ \ell) \cap \mathrm{di})) \cap \mathrm{di}$ in $\mathcal{N}(\mathrm{di}, \cap)$ and $e_4 = ((([\ell]^{-1} \circ \ell) - \mathrm{id}) \circ (([\ell]^{-1} \circ \ell) - \mathrm{id})) - \mathrm{id}$ in $\mathcal{N}(^{-1}, -)$. We have $[\![e_3]\!]_{\mathcal{T}_3} \neq \emptyset$ and $[\![e_4]\!]_{\mathcal{T}_3} \neq \emptyset$, while $[\![e_3]\!]_{\mathcal{T}_2} = [\![e_4]\!]_{\mathcal{T}_2} = \emptyset$. □

The proof of Proposition 3.7 relies on languages being able to distinguish two or three structurally equivalent children of a node. To do so, the proof uses expressions in the smallest language fragments possible that are k-subtree-reducible but not $k-1$-subtree-reducible, $k \in \{2, 3\}$. Hence, the classification provided in Proposition 3.7 is, in a sense, strict.

4. The power of diversity

Relation algebra expressions that do not utilize diversity can only inspect a local neighborhood around a given node. To study this in more detail, we first define the notion of locality:

Definition 4.1. A query q is *local* if there exists $k \geq 0$ such that, for every tree $\mathcal{T}$, and for all nodes m and n, $(m, n) \in [\![q]\!]_{\mathcal{T}}$ if and only if $(m, n) \in [\![q]\!]_{\mathcal{T}'}$, with $\mathcal{T}'$ the smallest subtree of $\mathcal{T}$ containing all nodes at distance at most k from the nearest common ancestor of m and n.

Note that *local queries* can only inspect distinct parts of a tree if they can navigate between these parts via a fixed number of edge steps (independent of the size of the tree). As such, our notion of locality is much more restrictive than typical notions of locality for first-order logic, e.g., Hanf-locality and Gaifman-locality [23]. A straightforward induction on the length of expressions yields the following:

Proposition 4.2. *Let* $\mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}, \cap, -\}$. *Every expression in* $\mathcal{N}(\mathcal{F})$ *is local.*

The query di is not local. Combined with Proposition 4.2, we derive

Corollary 4.3. *Let* $\mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}, \cap, -\}$. *Already on unlabeled chains, we have* $\mathcal{N}(\mathrm{di}) \not\leq_{\mathrm{path}} \mathcal{N}(\mathcal{F})$.

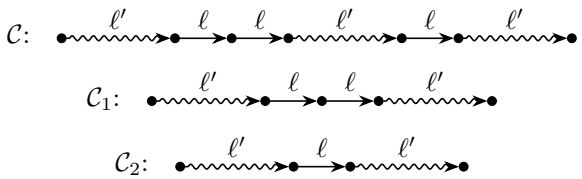

Figure 6: The chain $\mathcal{C}$ (*top*) can be distinguished from $\mathcal{C}_1$ (*middle*) and $\mathcal{C}_2$ (*bottom*) by detecting the presence of the patterns $\ell' \circ \ell \circ \ell'$ and $\ell' \circ \ell \circ \ell \circ \ell'$. The symbol $\rightsquigarrow$ represents a path of x edges, which can be used to make the paths sufficiently long for non-locality arguments. See the proof of Proposition 4.4 for details.

4.1. Adding diversity to local query languages

As already noticed, diversity always adds power to a local relation algebra fragment at the path level, as it can construct non-local relation algebra expressions. We can also use this property to our advantage to prove that diversity often adds expressive power at the Boolean level, too. To show this, we strengthen Corollary 4.3:

Proposition 4.4. *Let $\mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}, \cap, -\}$. Already on labeled chains, we have $\mathcal{N}(\mathrm{di}) \not\equiv_{\mathrm{bool}} \mathcal{N}(\mathcal{F})$.*

Proof. Consider the expression $e = \ell' \circ \ell \circ \ell' \circ \mathrm{di} \circ \ell' \circ \ell \circ \ell \circ \ell'$ and the chains $\mathcal{C}$, $\mathcal{C}_1$, and $\mathcal{C}_2$ of Figure 6. Independent of the length of the chains $\mathcal{C}$, $\mathcal{C}_1$, and $\mathcal{C}_2$, we have $[\![e]\!]_{\mathcal{C}} \neq \emptyset$, $[\![e]\!]_{\mathcal{C}_1} = \emptyset$, and $[\![e]\!]_{\mathcal{C}_2} = \emptyset$. Now, assume there exists an expression e' in $\mathcal{N}(\mathcal{F})$ with $e \equiv_{\mathrm{bool}} e'$. By Proposition 4.2, e' is local. Hence, we know there exists $k \geq 0$ such that e' satisfies Definition 4.1. Consider the chains $\mathcal{C}$, $\mathcal{C}_1$, and $\mathcal{C}_2$ of Figure 6 with $x > k$. Notice that every subchain of $\mathcal{C}$ containing all nodes at distance at most k from some given node is isomorphic to some subchain of either $\mathcal{C}_1$ or $\mathcal{C}_2$ of length at most $2k$. Hence, $(m, n) \in [\![e']\!]_{\mathcal{C}}$ implies either $[\![e']\!]_{\mathcal{C}_1} \neq \emptyset$, $[\![e']\!]_{\mathcal{C}_2} \neq \emptyset$, or both, contradicting $e \equiv_{\mathrm{bool}} e'$. Hence, no expression in $\mathcal{N}(\mathcal{F})$ is Boolean-equivalent to e. □

Next, we use the above techniques to prove a similar result for unlabeled trees:

Proposition 4.5. *Let $\mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}, \cap, -\}$. Already on unlabeled trees, we have $\mathcal{N}(\mathrm{di}, \cap) \not\equiv_{\mathrm{bool}} \mathcal{N}(\mathcal{F})$.*

Proof. Observe that $\underline{\{\mathrm{di}, \cap\}} = \{\mathrm{di}, \pi, \cap\}$ and consider the expression $e = P_{2,\neg r} \circ \mathrm{di} \circ P_{2,\neg r}$ with

$$\begin{aligned} P_{2,\neg r} &= \pi_2[\mathcal{E}] \circ P_2; \\ P_2 &= \pi_1[S_2]; \\ S_2 &= (\mathcal{E} \circ \mathrm{di}) \cap \mathcal{E}. \end{aligned}$$

The expression e is in $\mathcal{N}(\mathrm{di}, \pi, \cap)$ and selects node pairs among distinct non-root nodes such that each node in the pair has at least two distinct children. Now,

Figure 7: The trees $\mathcal{T}$ (*left*) and $\mathcal{T}'$ (*right*) can be distinguished by counting the number of non-root nodes that have two children, a non-local property. The symbol $\rightsquigarrow$ represents a path of x edges, which can be used to make the paths sufficiently long for non-locality arguments. See the proof of Proposition 4.5 for details.

assume there exists an expression e' in $\mathcal{N}(\mathcal{F})$ such that $e \equiv_{\text{bool}} e'$. Since e' is local, we know there exists $k \geq 0$ such that e' satisfies Definition 4.1. Consider the trees $\mathcal{T}$ and $\mathcal{T}'$ of Figure 7 with $x = 2k$. Clearly, $[\![e]\!]_{\mathcal{T}} \neq \emptyset$ and $[\![e]\!]_{\mathcal{T}'} = \emptyset$. By $e \equiv_{\text{bool}} e'$, we must have $[\![e']\!]_{\mathcal{T}} \neq \emptyset$. Let $(m, n) \in [\![e']\!]_{\mathcal{T}}$. Since every subtree of $\mathcal{T}$ containing all nodes at distance at most k from some given node is contained in a subtree of $\mathcal{T}$ that is isomorphic to $\mathcal{T}'$, we may conclude that $[\![e']\!]_{\mathcal{T}'} \neq \emptyset$. However, $[\![e]\!]_{\mathcal{T}'} = \emptyset$, contradicting $e \equiv_{\text{bool}} e'$. Hence, no expression in $\mathcal{N}(\mathcal{F})$ is Boolean-equivalent to e. □

4.2. Non-local query languages on chains

The erratic behavior of diversity in the non-local relation algebra fragments (allowing one to jump from any node to any other node in a tree) makes studying the expressive power of fragments with diversity inherently difficult. Luckily, we can obtain several separation results by studying these fragments on chains. The first set of separation results we prove use locality-based arguments on local subexpressions of non-local expressions, which allows us to prove limits on the expressive power of query languages that utilize diversity:

Proposition 4.6. *Let* $\mathcal{F} \subseteq \{\text{di}, {}^{-1}, \pi, \overline{\pi}, \cap, -\}$ *with* $\{\text{di}, {}^{-1}\} \subseteq \mathcal{F}$ *or* $\{\text{di}, \pi\} \subseteq \underline{\mathcal{F}}$. *Already on labeled chains, we have* $\mathcal{N}(\mathcal{F}) \not\equiv_{\text{bool}} \mathcal{N}(\text{di})$.

Proof. Consider the path-equivalent expressions

$$e_1 = (\ell \circ [\ell]^{-1}) \circ \text{di} \circ (\ell \circ [\ell]^{-1});$$
$$e_2 = \pi_1[\ell] \circ \text{di} \circ \pi_1[\ell].$$

The expression e_1 is in $\mathcal{N}(\text{di}, {}^{-1})$ and e_2 is in $\mathcal{N}(\text{di}, \pi)$. Choose $e \in \{e_1, e_2\}$. On the chains $\mathcal{C}$ and $\mathcal{C}'$ of Figure 8 we have $[\![e]\!]_{\mathcal{C}} \neq \emptyset$ and $[\![e]\!]_{\mathcal{C}'} = \emptyset$, independent of x, which will be determined next.

Now, assume there exists an expression e' in $\mathcal{N}(\text{di})$ such that $e \equiv_{\text{bool}} e'$. As id can always be removed from compositions, we may assume that e' is a union of $\{\text{id}, \cup\}$-free expressions.[4] Let S be the set of these $\{\text{id}, \cup\}$-free expressions in

[4]The exception would be the union-free expression id, but S cannot have such an union-free expression, as we assumed $e \equiv_{\text{bool}} e'$, and have $[\![e]\!]_{\mathcal{C}'} = \emptyset$.

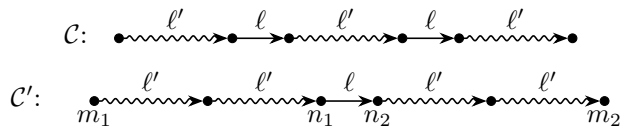

Figure 8: The chains $\mathcal{C}$ (*top*) and $\mathcal{C}'$ (*bottom*) can be distinguished by counting the number of edges labeled with ℓ, a non-local property. The symbol $\rightsquigarrow$ represents a path of x edges, which can be used to make the paths sufficiently long for non-locality arguments. See the proof of Proposition 4.6 for details.

e'. Choose x such that, for all $s \in S$, x is larger than the number of compositions in s.

Let $s \in S$ be an expression with $[\![s]\!]_{\mathcal{C}} \neq \emptyset$. Split s into a sequence of di-free expressions $t_1, \ldots, t_k$ such that $s = t_1 \circ \mathrm{di} \circ \ldots \circ \mathrm{di} \circ t_k$. First, we show that every *term* t_j, $1 \leq j \leq k$, is of the form ℓ'^y, with $0 \leq y < x$, or $\ell'^{z_1} \circ \ell \circ \ell'^{z_2}$, with $0 \leq z_1 + z_2 < x$ and $0 \leq z_1, z_2$. We do so by contradiction. We only have to consider the existence of a subexpressions of the form $\ell \circ \ell'^z \circ \ell$. By construction, $z < x$. Hence, $[\![\ell \circ \ell'^z \circ \ell]\!]_{\mathcal{C}} = \emptyset$ and $[\![s]\!]_{\mathcal{C}} = \emptyset$, a contradiction.

Let m_1 be the root of $\mathcal{C}'$, m_2 be the leaf of $\mathcal{C}'$, and $(n_1, n_2) \in [\![\ell]\!]_{\mathcal{C}'}$ be the ℓ-labeled edge in $\mathcal{C}'$ (see Figure 8). By induction on the number of terms $t_1, \ldots, t_i$ in s, $1 \leq i \leq k$, we shall prove that $[\![s]\!]_{\mathcal{C}'} \neq \emptyset$. The base cases are single-term expressions t_1. If t_1 is of the form ℓ'^y, then there exists a node m' in $\mathcal{C}'$ with $0 \leq \|m_1 \to m'\|_{\mathcal{C}'} = y < x$. Hence, $(m_1, m') \in [\![t_1]\!]_{\mathcal{C}'}$. Else, if t_1 is of the form $\ell'^{z_1} \circ \ell \circ \ell'^{z_2}$, then there exists nodes n'_1 and n'_2 with $0 \leq \|n'_1 \to n_1\|_{\mathcal{C}'} = z_1 < x$ and $0 \leq \|n_2 \to n'_2\|_{\mathcal{C}'} = z_2 < x$. Hence, $(n'_1, n'_2) \in [\![t_1]\!]_{\mathcal{C}'}$.

Now assume that, for all subexpressions $t_1 \circ \mathrm{di} \circ \ldots \circ \mathrm{di} \circ t_j$ of s with up-to-j di-free terms, $1 \leq j < i \leq k$, there exists a pair of nodes v, w with $(v, w) \in [\![t_1 \circ \mathrm{di} \circ \ldots \circ \mathrm{di} \circ t_j]\!]_{\mathcal{C}'}$ and either $0 \leq \|m_1 \to w\|_{\mathcal{C}'} < x$, $0 \leq \|n_2 \to w\|_{\mathcal{C}'} < x$, or $0 \leq \|w \to m_2\|_{\mathcal{C}'} < x$.

Next, we consider the subexpression $t_1 \circ \mathrm{di} \circ \ldots \circ \mathrm{di} \circ t_i$ of s with i di-free terms. Let $(v, w) \in [\![t_1 \circ \mathrm{di} \circ \ldots \circ \mathrm{di} \circ t_{i-1}]\!]_{\mathcal{C}'}$ such that v, w satisfy the statement of the induction hypothesis. Based on the form of t_i, we distinguish the following two cases:

1. If t_i is of the form ℓ'^y, then there exists nodes m'_1 and m'_2 with $0 \leq \|m_1 \to m'_1\|_{\mathcal{C}'} = y < x$ and $0 \leq \|m'_2 \to m_2\|_{\mathcal{C}'} = y < x$. Hence, $(m_1, m'_1) \in [\![t_i]\!]_{\mathcal{C}'}$ and $(m'_2, m_2) \in [\![t_i]\!]_{\mathcal{C}'}$. Due to the length of $\mathcal{C}$, $m'_1 \neq m'_2$. Hence, $w \neq m'_1$, $w \neq m'_2$, or both. Pick $m' \in \{m'_1, m'_2\}$ such that $w \neq m'$. We conclude $(v, m') \in [\![t_1 \circ \mathrm{di} \circ \ldots \circ \mathrm{di} \circ t_i]\!]_{\mathcal{C}'}$ and either $0 \leq \|m_1 \to m'\|_{\mathcal{C}'} < x$ or $0 \leq \|m' \to m_2\|_{\mathcal{C}'} < x$.
2. Else, if t_i is of the form $\ell'^{z_1} \circ \ell \circ \ell'^{z_2}$, then there exists nodes n'_1 and n'_2 with $0 \leq \|n'_1 \to n_1\|_{\mathcal{C}'} = z_1 < x$ and $0 \leq \|n_2 \to n'_2\|_{\mathcal{C}'} = z_2 < x$. Hence, $(n'_1, n'_2) \in [\![t_i]\!]_{\mathcal{C}'}$. Due to the length of $\mathcal{C}'$, we must have $n'_1 \neq w$ and we conclude $(v, n'_1) \in [\![t_1 \circ \mathrm{di} \circ \ldots \circ \mathrm{di} \circ t_i]\!]_{\mathcal{C}'}$ and $0 \leq \|n_2 \to n'_2\|_{\mathcal{C}'} < x$.

We conclude $[\![e']\!]_{\mathcal{C}'} \neq \emptyset$, a contradiction. Hence, no expression in $\mathcal{N}(\mathcal{F})$ is Boolean-equivalent to e. □

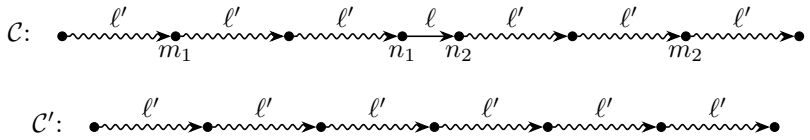

Figure 9: Chains $\mathcal{C}$ (*top*) and $\mathcal{C}'$ (*bottom*) can be distinguished by detecting the presence of an edge labeled ℓ, which is only present in chain $\mathcal{C}$. This property is non-local to most nodes in the chain, however (e.g., the root and the leaf), as the edge labeled ℓ is far removed from these nodes. The symbol $\rightsquigarrow$ represents a path of x edges, which can be used to make the paths sufficiently long for non-locality arguments. See the proof of Proposition 4.7 for details.

The above techniques can also be used with path queries, as shown next.

Proposition 4.7. *Let $\mathcal{F} \subseteq \{\mathrm{di}, ^{-1}\}$. Already on labeled chains, we have $\mathcal{N}(\mathrm{di}, ^{-1}, \pi) \not\equiv_{\mathrm{path}} \mathcal{N}(\mathcal{F})$.*

Proof. Consider the expression $e = \pi_1[(\mathrm{id} \cup \mathrm{di}) \circ \ell]$ and chains $\mathcal{C}$ and $\mathcal{C}'$ of Figure 9 with nodes m_1, m_2, n_1, and n_2 in $\mathcal{C}$ as illustrated. We have $[\![e]\!]_{\mathcal{C}} = [\![\mathrm{id}]\!]_{\mathcal{C}}$ and we have $[\![e]\!]_{\mathcal{C}'} = \emptyset$.

Now, assume there exists an expression e' in $\mathcal{N}(\mathcal{F})$ such that $e \equiv_{\mathrm{path}} e'$. As id can always be removed from compositions, we may assume that e' is a union of $\{\mathrm{id}, \cup\}$-free expressions.[5] Let S be the set of these $\{\mathrm{id}, \cup\}$-free expressions in e'. Choose x such that, for all $s \in S$, x is larger than the number of compositions in s.

Let $s \in S$ be an expression with $(m_1, m_1) \in [\![s]\!]_{\mathcal{C}}$. First, we prove by contradiction that s is not ℓ-free. Therefore, assume that s is ℓ-free and let s' be the expression obtained from s by replacing di by ℓ'. By construction, we have $[\![s']\!]_{\mathcal{C}'} \subseteq [\![s]\!]_{\mathcal{C}'}$. Observe that s' is a composition of at-most x terms of the form ℓ' and $[\ell']^{-1}$. As chain $\mathcal{C}'$ has a path of at-least x edges labeled with ℓ', we conclude $[\![s']\!]_{\mathcal{C}'} \neq \emptyset$, a contradiction. Hence, s is not ℓ-free.

Let $s = t_1 \circ \ldots \circ t_k$, with every t_i, $1 \leq i \leq k$, a term of the form ℓ, $[\ell]^{-1}$, ℓ', $[\ell']^{-1}$, or di. Let t_j, $1 \leq j \leq k$, be the last term of the form ℓ or $[\ell]^{-1}$. Observe that $\|m_1 \to n_1\|_{\mathcal{C}} = 2x$ and $\|m_1 \to n_2\|_{\mathcal{C}} = 2x + 1$. Hence, at least one of the terms t_i, $j < i \leq k$, must be di. Choose t_i, $j < i \leq k$, to be the first such term. Let $s'' = t_1 \circ \ldots \circ t_i \circ t'_{i+1} \circ \ldots \circ t'_k$ be the expression obtained from s by replacing all occurrences of di in $t_{i+1} \circ \ldots \circ t_k$ by ℓ'. By construction, we have $[\![t'_{i+1} \circ \ldots \circ t'_k]\!]_{\mathcal{C}} \subseteq [\![t_{i+1} \circ \ldots \circ t_k]\!]_{\mathcal{C}}$. Since $t'_{i+1} \circ \ldots \circ t'_k$ is a composition of at-most $x - 1$ terms of the form ℓ' and $[\ell']^{-1}$, there must exist a node m' with $-x < \|m' \to m_2\|_{\mathcal{C}} < x$ such that $(m', m_2) \in [\![t'_{i+1} \circ \ldots \circ t'_k]\!]_{\mathcal{C}}$. As $(m_1, m_1) \in [\![s]\!]_{\mathcal{C}}$, we must have $[\![t_j \circ \ldots \circ t_{i-1}]\!]_{\mathcal{C}} \neq \emptyset$. Notice that t_j is of the form ℓ or $[\ell]^{-1}$ and $t_{j+1} \circ \ldots \circ t_{i-1}$ is a composition of at most $x - 1$ terms of the form ℓ' and $[\ell']^{-1}$. Hence, if t_j is of the form ℓ, then there must exist a node w with $(n_1, w) \in [\![t_j \circ t_{j+1} \circ \ldots \circ t_{i-1}]\!]_{\mathcal{C}}$ and $1 \leq \|n_1 \to w\|_{\mathcal{C}} < x$.

[5] The exception would be the union-free expression id, but S cannot have such an union-free expression, as we assumed $e \equiv_{\mathrm{path}} e'$, and have $[\![e]\!]_{\mathcal{C}'} = \emptyset$.

Otherwise, if t_j is of the form $[\ell]^{-1}$, then there must exist a node w with $(n_2, w) \in [\![t_j \circ t_{j+1} \circ \ldots \circ t_{i-1}]\!]_{\mathcal{C}}$ and $-x < \|n_2 \to w\|_{\mathcal{C}} \leq -1$. In either case, we have $(m_1, w) \in [\![t_1 \circ \ldots \circ t_{i-1}]\!]_{\mathcal{C}}$ and, by construction and the length of $\mathcal{C}$, we have $w \neq m'$. Hence, $(m_1, m_2) \in [\![t_1 \circ \ldots \circ t_{i-1} \circ \mathrm{di} \circ t'_{i+1} \circ \ldots \circ t'_k]\!]_{\mathcal{C}}$ and we conclude $(m_1, m_2) \in [\![s]\!]_{\mathcal{C}}$, a contradiction. Hence, no expression in $\mathcal{N}(\mathcal{F})$ is path-equivalent to e. □

On chains, we have $\mathrm{di} \equiv_{\mathrm{path}} [\mathcal{E}]^+ \cup [[\mathcal{E}]^{-1}]^+$. Hence, on chains diversity can be expressed using the *ancestor axis* and the *descendant axis*, which in turn allows us to simplify projection terms which contain diversity, as we show next.

Lemma 4.8. *Let $\mathcal{F} \subseteq \{\mathrm{di}, {}^{-1}, \pi\}$ and let $\pi_j[e]$, $j \in \{1, 2\}$, be an expression in $\mathcal{N}(\mathcal{F})$. There exists a finite set S of expressions of the form $\pi_1[\mathcal{E}^v] \circ \pi_2[\mathcal{E}^w]$, $v, w \geq 0$, such that, on unlabeled chains, $\pi_j[e] \equiv_{\mathrm{path}} \bigcup S$.*

Proof. We have $\mathrm{di} \equiv_{\mathrm{path}} [\mathcal{E}]^+ \cup [[\mathcal{E}]^{-1}]^+$ and we also have $\pi_2[e] \equiv_{\mathrm{path}} \pi_1[[e]^{-1}]$. Hence, every projection expression $\pi_j[e]$, $j \in \{1, 2\}$, can be written as a union of expressions of the form $\pi_1[e']$ in which e' is build over the atoms id, $\mathcal{E}$, $[\mathcal{E}]^{-1}$, $[\mathcal{E}]^+$, and $[[\mathcal{E}]^{-1}]^+$, using the operators $\circ$ and π_1. We shall call such expressions e' *normal* in the remainder of this proof. So, it remains to show that Lemma 4.8 holds for expressions $\pi_1[e']$, with e' normal. We do this by structural induction on e'. We have the following base cases:

$$\pi_1[\mathrm{id}] \equiv_{\mathrm{path}} \mathrm{id} \equiv_{\mathrm{path}} \pi_1[\mathcal{E}^0] \circ \pi_2[\mathcal{E}^0];$$
$$\pi_1[\mathcal{E}] \equiv_{\mathrm{path}} \pi_1[[\mathcal{E}]^+] \equiv_{\mathrm{path}} \pi_1[\mathcal{E}^1] \circ \pi_2[\mathcal{E}^0];$$
$$\pi_1[[\mathcal{E}]^{-1}] \equiv_{\mathrm{path}} \pi_1[[[\mathcal{E}]^{-1}]^+] \equiv_{\mathrm{path}} \pi_1[\mathcal{E}^0] \circ \pi_2[\mathcal{E}^1].$$

Now, assume that Lemma 4.8 already holds for expressions $\pi_1[e'']$, with e'' a normal expression containing at most i operators, $i \geq 1$, and let $e = \pi_1[e']$ be an expression with e' a normal expression containing $i + 1$ operators. Then either $e' = \pi_1[e'_1]$ or $e' = e'_1 \circ e'_2$, with e'_1 and e'_2 normal expressions containing at most i operators. In the first case, $e = \pi_1[\pi_1[e'_1]] \equiv_{\mathrm{path}} \pi_1[e'_1]$, and Lemma 4.8 holds for e by the induction hypothesis. In the second case, we have $e = \pi_1[e'_1 \circ e'_2] \equiv_{\mathrm{path}} \pi_1[e'_1 \circ \pi_1[e'_2]]$. By the induction hypothesis, e'_2 is path-equivalent to a finite union of expressions of the form $\pi_1[\mathcal{E}^{v_2}] \circ \pi_2[\mathcal{E}^{w_2}]$, $v_2, w_2 \geq 0$. For e'_1, we distinguish again two cases:

1. Expression $e'_1 = \pi_1[e''_1]$, with e''_1 a normal expression containing at most i operators. Hence, by the induction hypothesis, e'_1 is path-equivalent to a finite union of expressions of the form $\pi_1[\mathcal{E}^{v_1}] \circ \pi_2[\mathcal{E}^{w_1}]$, $v_1, w_1 \geq 0$. It now suffices to observe that

   $$\pi_1[\mathcal{E}^{v_1}] \circ \pi_2[\mathcal{E}^{w_1}] \circ \pi_1[\mathcal{E}^{v_2}] \circ \pi_2[\mathcal{E}^{w_2}] \equiv_{\mathrm{path}} \pi_1[\mathcal{E}^{\max(v_1, v_2)}] \circ \pi_2[\mathcal{E}^{\max(w_1, w_2)}]$$

   to see that Lemma 4.8 holds for e.
2. In the other case, we may assume without loss of generality that e'_1 is an atom. Hence, it suffices to observe that, for $v, w \geq 0$,

   $$\pi_1[\mathrm{id} \circ (\pi_1[\mathcal{E}^v] \circ \pi_2[\mathcal{E}^w])] \equiv_{\mathrm{path}} \pi_1[\mathcal{E}^v] \circ \pi_2[\mathcal{E}^w];$$
   $$\pi_1[\mathcal{E} \circ (\pi_1[\mathcal{E}^v] \circ \pi_2[\mathcal{E}^w])] \equiv_{\mathrm{path}} \pi_1[\mathcal{E}^{v+1}] \circ \pi_2[\mathcal{E}^{\max(0, w-1)}];$$

$$\pi_1[[\mathcal{E}]^{-1} \circ (\pi_1[\mathcal{E}^v] \circ \pi_2[\mathcal{E}^w])] \equiv_{\text{path}} \pi_1[\mathcal{E}^{\max(0,v-1)}] \circ \pi_2[\mathcal{E}^{w+1}];$$
$$\pi_1[[\mathcal{E}]^{+} \circ (\pi_1[\mathcal{E}^v] \circ \pi_2[\mathcal{E}^w])] \equiv_{\text{path}} \bigcup_{1 \le i \le \max(1,w)} \pi_1[\mathcal{E}^{v+i}] \circ \pi_2[\mathcal{E}^{\max(0,w-i)}];$$
$$\pi_1[[[\mathcal{E}]^{-1}]^{+} \circ (\pi_1[\mathcal{E}^v] \circ \pi_2[\mathcal{E}^w])] \equiv_{\text{path}} \bigcup_{1 \le i \le \max(1,v)} \pi_1[\mathcal{E}^{\max(0,v-i)}] \circ \pi_2[\mathcal{E}^{w+i}],$$

to see that Lemma 4.8 holds for e in this case, too. □

Next, we illustrate the usage of the above technical lemma.

Example 4.9. Consider the expression $e = \pi_1[\text{di} \circ \mathcal{E} \circ \mathcal{E}]$. We have

$$\begin{aligned} e &\equiv_{\text{path}} \pi_1[[\mathcal{E}]^{+} \circ \pi_1[\mathcal{E}^2] \circ \pi_2[\mathcal{E}^0]] \cup \pi_1[[[\mathcal{E}]^{-1}]^{+} \circ \pi_1[\mathcal{E}^2] \circ \pi_2[\mathcal{E}^0]] \\ &\equiv_{\text{path}} \pi_1[\pi_1[\mathcal{E}^3] \circ \pi_2[\mathcal{E}^0]] \cup \pi_1[\pi_1[\mathcal{E}^1] \circ \pi_2[\mathcal{E}^1]] \cup \pi_1[\pi_1[\mathcal{E}^0] \circ \pi_2[\mathcal{E}^2]] \\ &\equiv_{\text{path}} \pi_1[\mathcal{E}^3] \circ \pi_2[\mathcal{E}^0] \cup \pi_1[\mathcal{E}^1] \circ \pi_2[\mathcal{E}^1] \cup \pi_1[\mathcal{E}^0] \circ \pi_2[\mathcal{E}^2]. \end{aligned}$$

As Example 4.9 shows, we can use Lemma 4.8 to partially eliminate diversity from non-local expressions on unlabeled chains. Using Lemma 4.8, we can also partially eliminate diversity from non-local expressions on chains, after which we can use additional distance-based arguments on subexpressions, as shown next.

Proposition 4.10. *Already on unlabeled chains, we have*

1. $\mathcal{N}(\text{di}, \cap) \not\equiv_{\text{path}} \mathcal{N}(\text{di}, {}^{-1}, \pi)$,
2. $\mathcal{N}({}^{-1}) \not\equiv_{\text{path}} \mathcal{N}(\text{di}, \pi)$.

Proof. First, we prove Statement 1. Consider the expression $e = (\text{di} \circ \mathcal{E}) \cap \text{di}$ in $\mathcal{N}(\text{di}, \cap)$. On a chain, this expression yields all pairs of distinct non-root nodes that are not edges. Now, assume there exists an expression e' in $\mathcal{N}(\text{di}, {}^{-1}, \pi)$ such that, on unlabeled chains, $e \equiv_{\text{path}} e'$. Since e is non-local, e' must be non-local too. Hence, e' must contain diversity. By Lemma 4.8, we can rewrite e' into a union of expressions each of which is a composition of terms of the form id, di, $\mathcal{E}$, $[\mathcal{E}]^{-1}$, $\pi_1[\mathcal{E}^v]$, or $\pi_2[\mathcal{E}^w]$, $v, w \ge 0$. Let $s = t_1 \circ \cdots \circ t_n$ be such an expression in which at least one term is diversity. By $\text{di} \equiv_{\text{path}} [\mathcal{E}]^{+} \cup [[\mathcal{E}]^{-1}]^{+}$, s is path-equivalent to the infinite union

$$\bigcup_{k_1,\ldots,k_n \neq 0} t_{1,k_1} \circ \cdots \circ t_{n,k_n},$$

in which $t_{i,k_i} = t_i$ if $t_i \neq \text{di}$ and $t_{i,k_i} = \mathcal{E}^{k_i}$ if $t_i = \text{di}$, $1 \le i \le n$. Each such term $\mathcal{E}^{k_i}$ is itself a composition of terms $\mathcal{E}$ (if $k_i > 0$) or of terms $[\mathcal{E}]^{-1}$ (if $k_i < 0$). Hence, each expression $s_{k_1,\ldots,k_n} = t_{1,k_1} \circ \cdots \circ t_{n,k_n}$ is itself a composition of terms of the form id, $\mathcal{E}$, $[\mathcal{E}]^{-1}$, $\pi_1[\mathcal{E}^v]$, or $\pi_2[\mathcal{E}^w]$. Let $\delta(s_{k_1,\ldots,k_n})$ and $\delta^{-1}(s_{k_1,\ldots,k_n})$ be the number of terms of the form $\mathcal{E}$ and $[\mathcal{E}]^{-1}$ in $s_{k_1,\ldots,k_n}$, respectively. We note that, for every chain $\mathcal{C}$, we have $(v, w) \in [\![s_{k_1,\ldots,k_n}]\!]_{\mathcal{C}}$ implies $\|v \to w\|_{\mathcal{C}} = \delta(s_{k_1,\ldots,k_n}) - \delta^{-1}(s_{k_1,\ldots,k_n})$. Since s contains at least one diversity term, the set

$$\{\delta(s_{k_1,\ldots,k_n}) - \delta^{-1}(s_{k_1,\ldots,k_n}) \mid k_1, \ldots, k_n \neq 0\}$$

covers all integer numbers with at most one exception (if s contains exactly one di term). We can therefore choose an expression $s' = s_{k_1,\ldots,k_n}$ for which $0 \leq \delta(s') - \delta^{-1}(s') \leq 1$. Observe that s' is a local expression in $\mathcal{N}(^{-1}, \pi)$. Now, choose an unlabeled chain $\mathcal{C}$ that is sufficiently long to ensure that $[\![s']\!]_{\mathcal{C}} \neq \emptyset$ (with at-least $\max(\delta(s'), \delta^{-1}(s'))$ edges). Then, $[\![s']\!]_{\mathcal{C}}$ contains either an identical node pair (if $\delta(s') - \delta^{-1}(s') = 0$) or an edge (if $\delta(s') - \delta^{-1}(s') = 1$), contradicting $e \equiv_{\text{path}} e'$. Hence, no expression in $\mathcal{N}(\text{di}, ^{-1}, \pi)$ is path-equivalent to e.

Next, we prove Statement 2 using a similar argument. Let $e = [\mathcal{E}]^{-1}$. Assume there exists an expression e' in $\mathcal{N}(\text{di}, \pi)$ such that, on unlabeled chains, $e \equiv_{\text{path}} e'$. Using the analysis from the previous case, we rewrite e' into an infinite union of expressions of the form $t_{1,k_1} \circ \cdots \circ t_{n,k_n}$ with $k_1, \ldots, k_n \neq 0$. Again, one can choose an expression $s' = t_{1,k_1} \circ \cdots \circ t_{n,k_n}$ for which $\delta(s') - \delta^{-1}(s') = 0$ or $\delta(s') - \delta^{-1}(s') = 1$. Hence, we conclude that e' is not path-equivalent to e. □

5. Brute-force results

In the above, we have presented the broad classification and separation results visualized in Figure 3. Several remaining separations, both at the path and Boolean levels, can be solved using brute-force techniques in the style of Fletcher et al. [18].

Consider two query languages $\mathcal{N}(\mathcal{F}_1)$ and $\mathcal{N}(\mathcal{F}_2)$ and class of graphs $\mathbb{G}$. By Definition 2.6, we do *not* have $\mathcal{N}(\mathcal{F}_1) \preceq_{\text{path},\mathbb{G}} \mathcal{N}(\mathcal{F}_2)$ on $\mathbb{G}$ if there exists an expression e_1 in $\mathcal{N}(\mathcal{F}_1)$ and a graph $\mathcal{G} \in \mathbb{G}$ such that, for all expressions e_2 in $\mathcal{N}(\mathcal{F}_2)$, we have $[\![e_1]\!]_{\mathcal{G}} \neq [\![e_2]\!]_{\mathcal{G}}$. From this observation, we derive a three-step way to determine whether $\mathcal{N}(\mathcal{F}_1) \not\preceq_{\text{path},\mathbb{G}} \mathcal{N}(\mathcal{F}_2)$:

1. First, select a graph $\mathcal{G} \in \mathbb{G}$ and an expression e_1 in $\mathcal{N}(\mathcal{F}_1)$.
2. Then compute the set $R = \{[\![e_2]\!]_{\mathcal{G}} \mid e_2 \text{ in } \mathcal{N}(\mathcal{F}_2)\}$.
3. Finally, if $[\![e_1]\!]_{\mathcal{G}} \notin R$, then we can conclude $\mathcal{N}(\mathcal{F}_1) \not\preceq_{\text{path},\mathbb{G}} \mathcal{N}(\mathcal{F}_2)$.

To compute the set R at the second step, we can use a brute-force method.

At the core of the brute-force computation of R is the observation that there is only a finite number of possible evaluation results on a given graph, even though there are an infinite number of expressions. Consequently, one can effectively enumerate all possible evaluation results on a given graph. To do so, one starts with evaluating all atomic expressions in $\mathcal{N}(\mathcal{F}_2)$, namely, the expressions of the form $\emptyset$, id, di (if di $\in \mathcal{F}_2$), ℓ (for every $\ell \in \Sigma$), and $[\ell]^{-1}$ (if $^{-1} \in \mathcal{F}_2$), yielding an initial set R of evaluation results. Next, we iteratively add evaluation results for non-atomic expressions. First, we consider the unary operators π and $\overline{\pi}$. To do so, we consider every new evaluation result r we produce. Let e_r be the expression used to produce r (hence, $r = [\![e_r]\!]_{\mathcal{G}}$) and let f_j with $f \in \{\pi, \overline{\pi}\}$ and $j \in \{1, 2\}$ be a unary operator. If $f_j \in \mathcal{F}_2$, then we check whether the evaluation $[\![f_j[e_r]]\!]_{\mathcal{G}}$ produces a new result. If so, we add this new result to R. Finally, we consider the Boolean operators $\circ$, $\cup$, $\cap$, and $-$. To do so, we consider every pair of evaluation results r_1 and r_2 we produce. Let $e_{r,1}$ and $e_{r,2}$ be the expression used to produce r_1 and r_2 (hence, $r_1 = [\![e_{r,1}]\!]_{\mathcal{G}}$ and $r_2 = [\![e_{r,2}]\!]_{\mathcal{G}}$) and

Figure 10: The unlabeled tree $\mathcal{T}$ and chain $\mathcal{C}$ used by the brute-force procedure to determine path separations.

let $\otimes \in \{\circ, \cup, \cap, -\}$ be a binary operator. If $\otimes \in \{\circ, \cup\}$ or $\otimes \in \mathcal{F}_2$, then we check whether the evaluation $[\![e_{r,1} \otimes e_{r,2}]\!]_{\mathcal{G}}$ produces a new result. If so, we add this new result to R. Eventually, no new evaluation results will be found, at which point we obtained R. The implementation of our brute-force tools can be found at `http://www.jhellings.nl/projects/bruteforce/` and we refer to Hellings [28] for further details.

Using this approach, we prove the following.

Proposition 5.1. *Let* $\mathcal{F}_1 \subseteq \{\mathrm{di}, \pi, \overline{\pi}, \cap, -\}$, $\mathcal{F}_2 \subseteq \{\mathrm{di}, ^{-1}\}$, *and let* $\mathcal{F}_3 \subseteq \{\mathrm{di}\}$.

1. *Already on unlabeled trees, we have* $\mathcal{N}(^{-1}) \not\preceq_{\mathrm{path}} \mathcal{N}(\mathcal{F}_1)$.
2. *Already on unlabeled trees, we have* $\mathcal{N}(\pi) \not\preceq_{\mathrm{path}} \mathcal{N}(\mathcal{F}_2)$.
3. *Already on unlabeled chains, we have* $\mathcal{N}(\pi) \not\preceq_{\mathrm{path}} \mathcal{N}(\mathcal{F}_3)$.

Proof. Let $\mathcal{T}$ and $\mathcal{C}$ be the tree and chain of Figure 10. Consider the expressions $e_1 = [\mathcal{E}]^{-1}$, $e_2 = \pi_1[\mathcal{E}] \circ \pi_2[\mathcal{E}]$, and $e_3 = \pi_1[\mathcal{E} \circ \mathcal{E}]$. An exhaustive brute-force search shows that no expression in $\mathcal{N}(\mathcal{F}_1)$ evaluates to $[\![e_1]\!]_{\mathcal{T}}$ on $\mathcal{T}$, no expression in $\mathcal{N}(\mathcal{F}_2)$ evaluates to $[\![e_2]\!]_{\mathcal{T}}$ on $\mathcal{T}$, and no expression in $\mathcal{N}(\mathcal{F}_3)$ evaluates to $[\![e_3]\!]_{\mathcal{C}}$ on $\mathcal{C}$. □

At the Boolean level, the key notion in the brute-force approach is the ability to *distinguish* a pair of graphs. Consider a class of graphs $\mathbb{G}$, and graphs $\mathcal{G}_1, \mathcal{G}_2 \in \mathbb{G}$. We say that a query q *distinguishes* $\mathcal{G}_1$ and $\mathcal{G}_2$ if either $[\![q]\!]_{\mathcal{G}_1} = \emptyset \neq [\![q]\!]_{\mathcal{G}_2}$ or $[\![q]\!]_{\mathcal{G}_1} \neq \emptyset = [\![q]\!]_{\mathcal{G}_2}$. We say that query language $\mathcal{N}(\mathcal{F}_1)$ can *distinguish* $\mathcal{G}_1$ and $\mathcal{G}_2$ if there exists an expression e_1 in $\mathcal{N}(\mathcal{F}_1)$ that distinguishes $\mathcal{G}_1$ and $\mathcal{G}_2$. Consider a second query language $\mathcal{N}(\mathcal{F}_2)$. By Definition 2.6, we do *not* have $\mathcal{N}(\mathcal{F}_1) \preceq_{\mathrm{bool},\mathbb{G}} \mathcal{N}(\mathcal{F}_2)$ on $\mathbb{G}$ if $\mathcal{N}(\mathcal{F}_1)$ can distinguish graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ while $\mathcal{N}(\mathcal{F}_2)$ cannot distinguish these graphs. From this observation, we derive a three-step way to determine whether $\mathcal{N}(\mathcal{F}_1) \not\preceq_{\mathrm{bool},\mathbb{G}} \mathcal{N}(\mathcal{F}_2)$:

1. First, select a pair of graphs $\mathcal{G}_1, \mathcal{G}_2 \in \mathbb{G}$ and an expression e_1 in $\mathcal{N}(\mathcal{F}_1)$ such that e_1 distinguishes $\mathcal{G}_1$ and $\mathcal{G}_2$.
2. Then compute the set $R = \{([\![e_2]\!]_{\mathcal{G}_1}, [\![e_2]\!]_{\mathcal{G}_2}) \mid e_2 \text{ in } \mathcal{N}(\mathcal{F}_2)\}$.
3. Finally, if, for no pair $(r_1, r_2) \in R$, we have $r_1 = \emptyset \neq r_2$ or $r_1 \neq \emptyset = r_2$, then we can conclude $\mathcal{N}(\mathcal{F}_1) \not\preceq_{\mathrm{bool},\mathbb{G}} \mathcal{N}(\mathcal{F}_2)$.

To compute the set R at the second step, we can use a brute-force enumeration method similar to the brute-force method used to prove Proposition 5.1. Using this approach, we prove the following.

Proposition 5.2. *Let* $\mathcal{F}_1 \subseteq \{\mathrm{di}\}$, $\mathcal{F}_2 \subseteq \{\mathrm{di}, ^{-1}\}$, *and* $\mathcal{F}_3 \subseteq \{\mathrm{di}, \pi, \overline{\pi}, \cap, -\}$. *Already on unlabeled trees, we have*

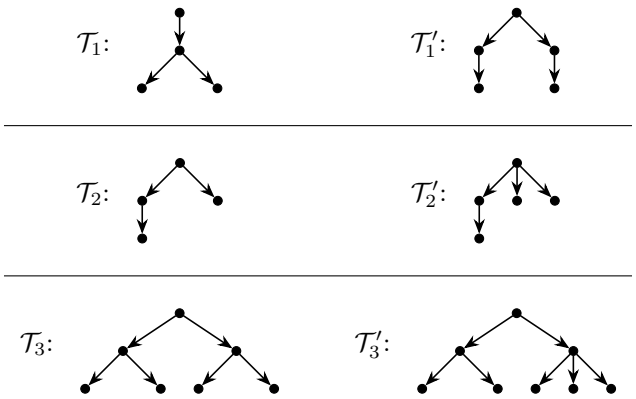

Figure 11: The pairs of unlabeled trees $(\mathcal{T}_1, \mathcal{T}_1')$, $(\mathcal{T}_2, \mathcal{T}_2')$, and $(\mathcal{T}_3, \mathcal{T}_3')$ used by the brute-force procedure to determine Boolean separations.

1. $\mathcal{N}(\mathrm{di}, {}^{-1}) \not\preceq_{\mathrm{bool}} \mathcal{N}(\mathcal{F}_1)$;
2. $\mathcal{N}(\mathrm{di}, \pi) \not\preceq_{\mathrm{bool}} \mathcal{N}(\mathcal{F}_2)$;
3. $\mathcal{N}(\mathrm{di}, {}^{-1}, \cap) \not\preceq_{\mathrm{bool}} \mathcal{N}(\mathcal{F}_3)$.

Proof. Let $\mathcal{T}_1$, $\mathcal{T}_1'$, $\mathcal{T}_2$, $\mathcal{T}_2'$, $\mathcal{T}_3$, and $\mathcal{T}_3'$ be the trees of Figure 11. Consider the following three expressions:

$$\begin{aligned} e_1 &= ([\mathcal{E}]^{-1} \circ [\mathcal{E}]^{-1} \circ \mathcal{E}) \circ \mathrm{di} \circ ([\mathcal{E}]^{-1} \circ \mathcal{E} \circ \mathcal{E}); \\ e_2 &= (\mathcal{E} \circ \mathcal{E} \circ [\mathcal{E}]^{-1}) \circ \mathrm{di} \circ \pi_1[[\mathcal{E}]^{-1} \circ \mathcal{E} \circ \mathcal{E}] \circ \mathrm{di} \\ &\qquad \circ \pi_1[[\mathcal{E}]^{-1} \circ \mathcal{E} \circ \mathcal{E}] \circ \mathrm{di} \circ (\mathcal{E} \circ [\mathcal{E}]^{-1} \circ [\mathcal{E}]^{-1}); \\ e_3 &= ((([\mathcal{E}]^{-1} \circ \mathcal{E}) \cap \mathrm{di}) \circ (([\mathcal{E}]^{-1} \circ \mathcal{E}) \cap \mathrm{di})) \cap \mathrm{di}. \end{aligned}$$

We have $[\![e_1]\!]_{\mathcal{T}_1} = \emptyset$ and $[\![e_1]\!]_{\mathcal{T}_1'} \neq \emptyset$. An exhaustive brute-force search shows that no expression in $\mathcal{N}(\mathcal{F}_1)$ can distinguish $\mathcal{T}_1$ from $\mathcal{T}_1'$. Next, we have $[\![e_2]\!]_{\mathcal{T}_2} = \emptyset$ and $[\![e_2]\!]_{\mathcal{T}_2'} \neq \emptyset$. Observe that e_2 is in $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi)$. By Proposition 8.4 (Section 8), e_2 is Boolean-equivalent to an expression in $\mathcal{N}(\mathrm{di}, \pi)$. An exhaustive search shows that no expression in $\mathcal{N}(\mathcal{F}_2)$ can distinguish $\mathcal{T}_2$ from $\mathcal{T}_2'$. Finally, we have $[\![e_3]\!]_{\mathcal{T}_2} = \emptyset$ and $[\![e_3]\!]_{\mathcal{T}_2'} \neq \emptyset$. An exhaustive search shows that no expression in $\mathcal{N}(\mathcal{F}_3)$ can distinguish $\mathcal{T}_3$ from $\mathcal{T}_3'$. □

6. Collapse results

In Sections 3, 4, and 5, we focused on separation results. Here, we focus on collapse results. To prove these collapse results, we introduce *condition tree queries*, a generalization of the tree queries of Wu et al. [14] (see also Section 8).

First, in Section 6.1, we introduce the condition tree queries and show that they can be used to represent union-free local relation algebra expression. Next, in Section 6.2, we show that the condition tree queries are closed under intersection, which we use to show that intersection is redundant for the local

relation algebra fragments. Finally, in Section 6.3, we use the condition tree queries to show the Boolean-equivalence of $\mathcal{N}(^{-1})$ and $\mathcal{N}(\pi)$.

6.1. Condition tree queries

We first define the syntax and semantics of condition tree queries:

Definition 6.1. A *condition tree query* $\mathcal{Q}$ is a tuple $\mathcal{Q} = (\mathcal{T}, C, \mathsf{s}, \mathsf{t}, \gamma)$, where $\mathcal{T} = (\mathcal{V}, \Sigma, \mathbf{E})$ is a labeled tree, C is a set of node expressions that represent *node conditions*, $\mathsf{s} \in \mathcal{V}$ is the *source node*, $\mathsf{t} \in \mathcal{V}$ is the *target node*, and $\gamma \subseteq \mathcal{V} \times C$ is the *node-condition relation*. We write $\gamma(n)$ to denote the set $\{e \mid (n, e) \in \gamma\}$.[6] Let $\mathcal{T}' = (\mathcal{V}', \Sigma, \mathbf{E}')$ be a tree. Then $[\![\mathcal{Q}]\!]_{\mathcal{T}'}$ consists of all node pairs $(m, n) \in \mathcal{V}' \times \mathcal{V}'$ for which there exists a mapping $f : \mathcal{V} \to \mathcal{V}'$ satisfying the following conditions:

1. $f(\mathsf{s}) = m$ and $f(\mathsf{t}) = n$;
2. for all $v \in \mathcal{V}$ and $e \in \gamma(v)$, $(f(v), f(v)) \in [\![e]\!]_{\mathcal{T}'}$; and
3. for all $\ell \in \Sigma$ and $(v, w) \in \mathbf{E}(\ell)$, $(f(v), f(w)) \in \mathbf{E}'(\ell)$.

We denote the condition tree query path-equivalent to $\emptyset$ simply by $\emptyset$.

Intuitively speaking, condition tree queries define intentional relationships between pairs of nodes (m, n) by specifying a tree pattern that connects source node m and target node n. In this tree pattern, each individual node must satisfy the specified node conditions (for that node in the pattern).

Example 6.2. The condition tree query $\mathcal{Q}$ in Figure 12 selects a node pair (s, t) if the following tree traversal steps are all successful:

1. from s, go up via two edges labeled ℓ_1 and ℓ_2;
2. check if the node where we have arrived satisfies the node expression $\overline{\pi}_2[\ell_3]$;
3. from there, go down via two edges labeled ℓ_3, arriving at t;
4. check if t has outgoing edges labeled ℓ_1 and ℓ_2.

The condition tree query $\mathcal{Q}$ is path-equivalent to the expression

$$[\ell_1]^{-1} \circ [\ell_2]^{-1} \circ \overline{\pi}_2[\ell_3] \circ \ell_3 \circ \ell_3 \circ \pi_1[\ell_1] \circ \pi_1[\ell_2],$$

which is in $\mathcal{N}(^{-1}, \overline{\pi})$.

In the remainder of this section, we formalize the relationship between condition tree queries and relation algebra expressions exhibited in Example 6.2. Thereto, let $\mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}\}$, and let $\mathbf{Q}_{\mathtt{tree}}(\mathcal{F})$ be the class of all condition tree queries in which node conditions are restricted to $\cup$-free expressions in $\mathcal{N}(\mathcal{F})$. We claim that, for $\underline{\mathcal{F}} = \{^{-1}, \pi\}$ and $\underline{\mathcal{F}} = \{^{-1}, \pi, \overline{\pi}\}$, $\mathbf{Q}_{\mathtt{tree}}(\mathcal{F})$ and the class of all $\cup$-free expressions in $\mathcal{N}(\mathcal{F})$ path-subsume each other.

First, we show how to rewrite from $\mathcal{N}(\mathcal{F})$ to a condition tree query:

[6] We observe that the condition tree queries without node conditions are equivalent to the tree queries as defined in Wu et al. [14].

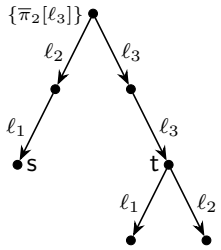

Figure 12: A condition tree query with a single condition.

Algorithm PRODUCETQ($t_1 \circ \cdots \circ t_k$, Σ)**:**

1: s := a fresh tree node
2: $\mathcal{V}, \mathbf{E}, C, \gamma(\mathsf{s}), \mathit{current} := \{\mathsf{s}\}, \{\ell \mapsto \emptyset \mid \ell \in \Sigma\}, \emptyset, \emptyset, \mathsf{s}$
3: **for** t_i in $[t_1, \ldots, t_k]$ **do**
4: **if** t_i is of the form ℓ **then**
5: n := a fresh tree node
6: $\mathcal{V}, \mathbf{E}(\ell), \mathit{current} := \mathcal{V} \cup \{n\}, \mathbf{E}(\ell) \cup \{(\mathit{current}, n)\}, n$
7: **else if** t_i is of the form $[\ell]^{-1}$ **then**
8: **if not** $\exists n \exists \ell'$ with $(n, \mathit{current}) \in \mathbf{E}(\ell')$ **then**
9: n := a fresh tree node
10: $\mathcal{V}, \mathbf{E}(\ell), \mathit{current} := \mathcal{V} \cup \{n\}, \mathbf{E}(\ell) \cup \{(n, \mathit{current})\}, n$
11: **else if** $\exists n \exists \ell'$ with $\ell = \ell'$ and $(n, \mathit{current}) \in \mathbf{E}(\ell')$ **then**
12: $\mathit{current} := n$
13: **else #**($\exists n \exists \ell'$ with $\ell \neq \ell'$ and $(n, \mathit{current}) \in \mathbf{E}(\ell')$).
14: **return** $\emptyset$
15: **else #**(t_i is a node expression).
16: $C, \gamma(\mathit{current}) := C \cup \{t_i\}, \gamma(\mathit{current}) \cup \{t_i\}$
17: **return** $((\mathcal{V}, \Sigma, \mathbf{E}), C, \mathsf{s}, \mathit{current}, \gamma)$

Figure 13: Algorithm PRODUCETQ that translates $\cup$-free expressions in $\mathcal{N}(^{-1}, \pi, \overline{\pi})$ to path-equivalent condition tree queries (given a set Σ of possible edge labels).

Proposition 6.3. *Let $\mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}\}$ and let e be a $\cup$-free expression in $\mathcal{N}(\mathcal{F})$. There exists a condition tree query $\mathcal{Q}$ in $\mathbf{Q}_{\mathtt{tree}}(\mathcal{F})$ such that $e \equiv_{\mathrm{path}} \mathcal{Q}$.*

Proof. Due to e being $\cup$-free and in $\mathcal{N}(^{-1}, \pi, \overline{\pi})$, it is of the form $t_1 \circ t_2 \circ \cdots \circ t_k$, in which every t_i, $1 \leq i \leq k$, is an expression of the form $\emptyset$, id, ℓ, $[\ell]^{-1}$, $\pi_j[e']$, or $\overline{\pi}_j[e']$, with ℓ an edge label and $j \in \{1, 2\}$. Algorithm PRODUCETQ, presented in Figure 13, will construct the path-equivalent condition tree query from $t_1 \circ \cdots \circ t_n$ that satisfies this Proposition. It is straightforward to prove that the main for-loop satisfies the loop invariant "the structure $((\mathcal{V}, \Sigma, \mathbf{E}), C, \mathsf{s}, \mathit{current}, \gamma)$ is a condition tree query in $\mathbf{Q}_{\mathtt{tree}}(\mathcal{F})$ and is path-equivalent to id $\circ\, t_1 \circ \cdots \circ t_{i-1}$." Hence, termination of the Algorithm yields the desired query $\mathcal{Q}$. □

For the translation from condition tree queries to $\cup$-free expressions, we shall

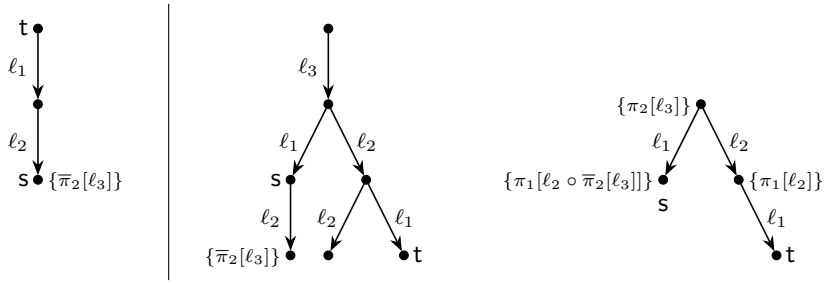

Figure 14: The condition tree query on the *left* is up-down. The condition tree query in the *middle* is not, but this query is path-equivalent to the up-down query on the right.

assume that the condition tree queries are restricted to *up-down queries*.

Definition 6.4. A condition tree query $\mathcal{Q} = (\mathcal{T}, C, \mathsf{s}, \mathsf{t}, \gamma)$ is an *up-down query* if all edges of $\mathcal{T}$ are on the unique path from s to t (not taking into account the direction of the edges).

Example 6.5. An up-down query can look like a chain if the target node is an ancestor of the source node, or vice versa, as illustrated by Figure 14, *left*. This up-down query is path-equivalent to $\overline{\pi}_2[\ell_3] \circ [\ell_2]^{-1} \circ [\ell_1]^{-1}$.

The condition tree query in the *middle* is not up-down, but is path-equivalent to the up-down tree query on the *right*. Observe that the *right* query is obtained by pushing parts of the tree traversal described by the *middle* query into node conditions. The *middle* and *right* queries are path-equivalent to $\pi_1[\ell_2 \circ \overline{\pi}_2[\ell_3]] \circ [\ell_1]^{-1} \circ \pi_2[\ell_3] \circ \ell_2 \circ \pi_1[\ell_2] \circ \ell_1$.

As illustrated in Example 6.5, we can rewrite a condition tree query to an up-down query by pushing into node conditions those parts of the condition tree query not on the path from source to target. The first step in this is to push all node conditions of a node into a single expression, for which we define the *condition expression*.

Definition 6.6. Let $((\mathcal{V}, \Sigma, \mathbf{E}), C, \mathsf{s}, \mathsf{t}, \gamma)$ be a condition tree query and let $n \in \mathcal{V}$. We define the *condition expression* $\mathsf{conditions}(n)$ of n by

$$\mathsf{conditions}(n) = \bigcap_{e \in \gamma(n)} e$$

when $\gamma(n) \neq \emptyset$ and $\mathsf{conditions}(n) = \mathrm{id}$ otherwise. As every expression in $\gamma(n)$ is a node expression, the expression $\mathsf{conditions}(n)$ is also a node expression. Observe that if $\gamma(n) = \{e_1, \ldots, e_k\}$, then $\mathsf{conditions}(n) \equiv_{\mathrm{path}} e_1 \circ \cdots \circ e_k$. We usually assume that $\mathsf{conditions}(n)$ is written as such a composition of terms instead of an intersection of terms.

Lemma 6.7. *Let $\{\pi\} \subseteq \underline{\mathcal{F}} \subseteq \{^{-1}, \overline{\pi}, \pi\}$ and let $\mathcal{Q}$ be a condition tree query in $\mathbf{Q}_{\mathtt{tree}}(\mathcal{F})$. There exists an up-down query $\mathcal{Q}'$ in $\mathbf{Q}_{\mathtt{tree}}(\mathcal{F})$ such that $\mathcal{Q} \equiv_{\mathrm{path}} \mathcal{Q}'$.*

Proof. Let $\mathcal{Q} = (\mathcal{T}, C, \mathsf{s}, \mathsf{t}, \gamma)$ with $\mathcal{T} = (\mathcal{V}, \Sigma, \mathbf{E})$. If $\mathcal{Q}$ is not up-down, then there must exist a node $n \in \mathcal{V} - \{\mathsf{s}, \mathsf{t}\}$ such that n is a leaf node or such that n is the root node and has only a single child. In both cases, we are able to remove n from $\mathcal{Q}$. First, we consider the *leaf-prune* of leaf node n. Let $(m, n) \in \mathbf{E}(\ell)$ be the edge in $\mathcal{T}$ that connects n to its parent m. Remove n from the tree and add the node expression $\pi_1[\ell \circ \mathsf{conditions}(n)]$ to C and $\gamma(m)$. Let $\mathcal{Q}''$ be the result of the leaf-prune. By construction, we have $\mathcal{Q}''$ in $\mathbf{Q}_{\mathtt{tree}}(\mathcal{F})$ and $\mathcal{Q} \equiv_{\text{path}} \mathcal{Q}''$. Next, we consider the *root-prune* of root node n. Let $(n, m) \in \mathbf{E}(\ell)$ be the edge in $\mathcal{T}$ that connects n to its single child m. Remove n from the tree and add the node expression $\pi_2[\mathsf{conditions}(n) \circ \ell]$ to C and $\gamma(m)$. Let $\mathcal{Q}''$ be the result of the root-prune. By construction, we have $\mathcal{Q}''$ in $\mathbf{Q}_{\mathtt{tree}}(\mathcal{F})$ and $\mathcal{Q} \equiv_{\text{path}} \mathcal{Q}''$. By repeatedly applying leaf-pruning and root-pruning until both are no longer possible, we end up with an up-down query that satisfies the Lemma. □

As illustrated in Example 6.5, an up-down query can be translated straightforwardly into a path-equivalent relation algebra expression, provided we have the converse operator $(^{-1})$ at our disposal:

Proposition 6.8. *Let $\{^{-1}\} \subseteq \mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}\}$ and let $\mathcal{Q}$ be a condition tree query in $\mathbf{Q}_{\mathtt{tree}}(\mathcal{F})$. There exists a $\cup$-free expression e in $\mathcal{N}(\mathcal{F})$ such that $e \equiv_{\text{path}} \mathcal{Q}$.*

Proof. Due to Lemma 6.7, we may assume, without loss of generality, that $\mathcal{Q} = (\mathcal{T}, C, \mathsf{s}, \mathsf{t}, \gamma)$ is an up-down query. Hence, we can represent $\mathcal{Q}$ by two paths $r\, \ell_{\mathsf{s},1}\, m_1 \cdots \ell_{\mathsf{s},u}\, \mathsf{s}$ and $r\, \ell_{\mathsf{t},1}\, n_1 \cdots \ell_{\mathsf{t},d}\, \mathsf{t}$. Observe that if $r = \mathsf{s}$, then the first path reduces to just r and if $r = \mathsf{t}$, then the second path reduces to just r. The expression

$$\mathsf{conditions}(\mathsf{s}) \circ [\ell_{\mathsf{s},u}]^{-1} \circ \cdots \circ \mathsf{conditions}(m_1) \circ [\ell_{\mathsf{s},1}]^{-1} \circ \mathsf{conditions}(r) \circ \ell_{\mathsf{t},1} \circ \mathsf{conditions}(n_1) \circ \cdots \circ \ell_{\mathsf{t},d} \circ \mathsf{conditions}(\mathsf{t})$$

is in $\mathcal{N}(\mathcal{F})$ and is path-equivalent to $\mathcal{Q}$. □

6.2. Adding intersection to local fragments

For path queries, Hellings et al. [12, 13] already proved that adding intersection to local relation algebra fragments not containing the converse operator (the *downward* relation algebra fragments) never increases their expressive power (Proposition 8.3 in Section 8). Here, we show that this result actually holds for *all* local relation algebra fragments. As a first step, consider the following example:

Example 6.9. Suppose we want to compute the intersection of the two up-down queries in Figure 15, *left*. Since the two up-down queries have different depths, a pair of nodes of a tree can only be in the result of the intersection of the two queries on that tree if the children of the root of the second query are mapped to the same node. Hence, we can replace the second up-down query by the one shown in the *middle*. Since both queries now have the same shape and corresponding edges have the same label, the intersection is easily obtained by merging the node conditions, resulting in the up-down query on the *right*.

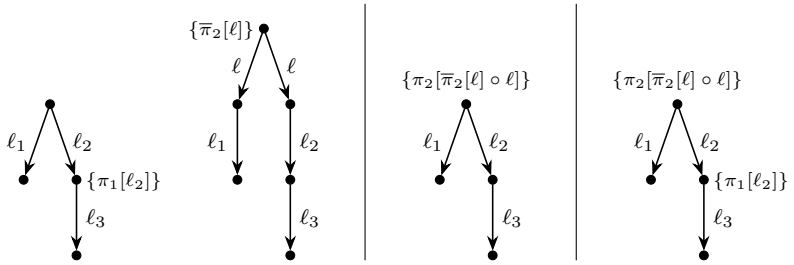

Figure 15: The step-by-step intersection of the two up-down queries on the *left*, resulting in the up-down query on the *right*.

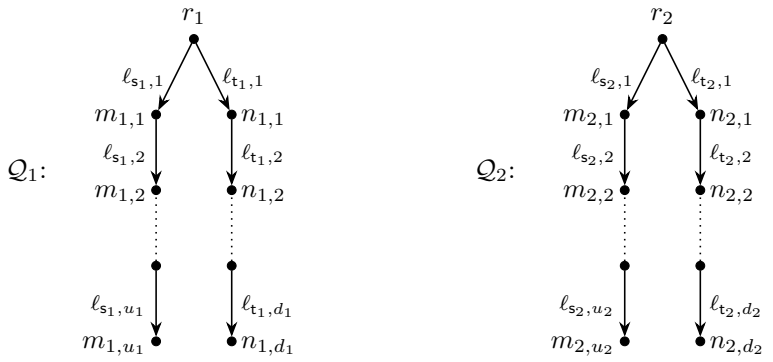

Figure 16: Up-down queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$, as used in the proof of Proposition 6.10.

Next, we show that the steps taken in Example 6.9 are sound:

Proposition 6.10. *Let $\{\pi\} \subseteq \mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}\}$ and let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be condition tree queries in $\mathbf{Q}_{\mathtt{tree}}(\mathcal{F})$. There exists a condition tree query $\mathcal{Q}$ such that, for every tree $\mathcal{T}$, $[\![\mathcal{Q}]\!]_{\mathcal{T}} = [\![\mathcal{Q}_1]\!]_{\mathcal{T}} \cap [\![\mathcal{Q}_2]\!]_{\mathcal{T}}$.*

Proof. Let $\mathcal{Q}_1 = (\mathcal{T}_1, C_1, \mathtt{s}_1, \mathtt{t}_1, \gamma_1)$ and $\mathcal{Q}_2 = (\mathcal{T}_2, C_2, \mathtt{s}_2, \mathtt{t}_2, \gamma_2)$ be condition tree queries. By Lemma 6.7, we can assume, without loss of generality, that $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are up-down queries. As $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are up-down queries, we can represent $\mathcal{Q}_1$ by two paths $r_1\, \ell_{\mathtt{s}_1,1}\, m_{1,1} \cdots \ell_{\mathtt{s}_1,u_1}\, \mathtt{s}_1$ and $r_1\, \ell_{\mathtt{t}_1,1}\, n_{1,1} \cdots \ell_{\mathtt{t}_1,d_1}\, \mathtt{t}_1$ and $\mathcal{Q}_2$ by two paths $r_2\, \ell_{\mathtt{s}_2,1}\, m_{2,1} \cdots \ell_{\mathtt{s}_2,u_2}\, \mathtt{s}_2$ and $r_2\, \ell_{\mathtt{t}_2,1}\, n_{2,1} \cdots \ell_{\mathtt{t}_2,d_2}\, \mathtt{t}_2$, as visualized by the two tree queries in Figure 16.

Let $\mathcal{T}'$ be an arbitrary tree. If $u_1 - d_1 \neq u_2 - d_2$, then $[\![\mathcal{Q}_1]\!]_{\mathcal{T}'} \cap [\![\mathcal{Q}_2]\!]_{\mathcal{T}'} = \emptyset$. If $u_1 - d_1 = u_2 - d_2$, then we distinguish two cases:

1. $u_1 \neq u_2$ *and* $d_1 \neq d_2$. By symmetry, assume $u_2 > u_1$. Let $\Delta = d_2 - d_1 = u_2 - u_1$. Now assume that $(v, w) \in [\![\mathcal{Q}_1]\!]_{\mathcal{T}'} \cap [\![\mathcal{Q}_2]\!]_{\mathcal{T}'}$. Consider the mapping f from $\mathcal{Q}_2$ to $\mathcal{T}'$ that shows $(v, w) \in [\![\mathcal{Q}_2]\!]_{\mathcal{T}'}$. Due to $(v, w) \in [\![\mathcal{Q}_1]\!]_{\mathcal{T}'}$, this mapping f must map the first Δ nodes on the paths $r_2 \ldots \mathtt{s}_2$ and $r_2 \ldots \mathtt{t}_2$ to the same Δ nodes in $\mathcal{T}'$. Hence, if for some i, $1 \leq i \leq \Delta$, $\ell_{\mathtt{s}_2,i} \neq \ell_{\mathtt{t}_2,i}$, then $[\![\mathcal{Q}_1]\!]_{\mathcal{T}'} \cap [\![\mathcal{Q}_2]\!]_{\mathcal{T}'} = \emptyset$. Thus, assume, for all i, $1 \leq i \leq \Delta$, that

$\ell_{\mathsf{s}_2,i} = \ell_{\mathsf{t}_2,i}$. We can embed this restriction on the nodes $m_{2,1}, \ldots, m_{2,\Delta}$ and $n_{2,1}, \ldots, n_{2,\Delta}$ by replacing $\mathcal{Q}_2$ by the up-down query $\mathcal{Q}_2'$ represented by two paths $r\, \ell_{\mathsf{s}_2,\Delta+1}\, m_{2,\Delta+1} \cdots \ell_{2,u_2}\, \mathsf{s}_2$ and $r\, \ell_{\mathsf{t}_2,\Delta+1}\, n_{2,\Delta+1} \cdots \ell_{\mathsf{t}_2,d_2}\, \mathsf{t}_2$ with

$$\gamma(r) = \pi_2[\mathsf{conditions}(r_2) \circ \ell_{\mathsf{s}_2,1} \circ \mathsf{conditions}(m_{2,1}) \circ \mathsf{conditions}(n_{2,1}) \circ \cdots \circ \ell_{\mathsf{s}_2,\Delta} \circ \mathsf{conditions}(m_{2,\Delta}) \circ \mathsf{conditions}(n_{2,\Delta})].$$

On $\mathcal{Q}_1$ and $\mathcal{Q}_2'$ the conditions of the next case apply.

2. $u_1 = u_2$ *and* $d_1 = d_2$. In this case, $[\![\mathcal{Q}_1]\!]_{\mathcal{T}'} \cap [\![\mathcal{Q}_2]\!]_{\mathcal{T}'} = \emptyset$ whenever $\ell_{\mathsf{s}_1,i} \neq \ell_{\mathsf{s}_2,i}$, $1 \leq i \leq u_1 = u_2$, or $\ell_{\mathsf{t}_1,i} \neq \ell_{\mathsf{t}_2,i}$, $1 \leq i \leq d_1 = d_2$. In all other cases, we define $\mathcal{Q} = (\mathcal{T}_1, C_1 \cup C_2, \mathsf{s}_1, \mathsf{t}_1, \gamma')$, in which

$$\begin{aligned}
\gamma'(\mathsf{s}_1) &= \gamma(\mathsf{s}_1) \cup \gamma(\mathsf{s}_2); \\
\gamma'(\mathsf{t}_1) &= \gamma(\mathsf{t}_1) \cup \gamma(\mathsf{t}_2); \\
\gamma'(r_1) &= \gamma(r_1) \cup \gamma(r_2); \\
\gamma'(m_{1,i}) &= \gamma_1(m_{1,i}) \cup \gamma_2(m_{2,i}) \qquad \text{with } 1 \leq i \leq u_1 = u_2; \\
\gamma'(n_{1,i}) &= \gamma_1(n_{1,i}) \cup \gamma_2(n_{2,i}) \qquad \text{with } 1 \leq i \leq d_1 = d_2.
\end{aligned}$$

□

We can use Proposition 6.10 directly to remove intersection at the top level in $\cup$-free expressions, this via a translation to condition tree queries. To remove other occurrences of intersection, a straightforward induction argument on the structure of expressions suffices.

Theorem 6.11. *Let $\{^{-1}, \pi\} \subseteq \mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}, \cap\}$. On labeled trees, every $\cup$-free expression in $\mathcal{N}(\mathcal{F})$ is path-equivalent to a $\cup$-free expression in $\mathcal{N}(\mathcal{F} - \{\cap\})$.*

Proof. The proof is by induction on the length of e. The base cases are the atoms, which are all already in $\mathcal{N}(\mathcal{F} - \{\cap\})$. We assume that, for all $\cup$-free expressions e' in $\mathcal{N}(\mathcal{F})$ of length at most i, there exists a $\cup$-free expression e'' in $\mathcal{N}(\mathcal{F} - \{\cap\})$ such that $e' \equiv_{\text{path}} e''$.

Let e be a $\cup$-free expression in $\mathcal{N}(\mathcal{F})$ and not in $\mathcal{N}(\mathcal{F} - \{\cap\})$ of length $i + 1$. We distinguish the following three cases:

1. $e = e_1 \circ e_2$. We use the inductive hypothesis on e_1 and e_2 to obtain $\cup$-free expressions e_1' and e_2' in $\mathcal{N}(\mathcal{F} - \{\cap\})$ with $e_1 \equiv_{\text{path}} e_1'$ and $e_2 \equiv_{\text{path}} e_2'$. Hence, we have $e \equiv_{\text{path}} e_1' \circ e_2'$.
2. $e = f_j[e']$ *with* $f \in \{\pi, \overline{\pi}\}$ *and* $j \in \{1, 2\}$. We use the induction hypothesis on e' to obtain a $\cup$-free expression e'' in $\mathcal{N}(\mathcal{F} - \{\cap\})$ with $e' \equiv_{\text{path}} e''$. Hence, we have $e \equiv_{\text{path}} f_j[e'']$.
3. $e = e_1 \cap e_2$. We use the inductive hypothesis on e_1 and e_2 to obtain $\cup$-free expressions e_1' and e_2' in $\mathcal{N}(\mathcal{F} - \{\cap\})$ with $e_1 \equiv_{\text{path}} e_1'$ and $e_2 \equiv_{\text{path}} e_2'$. By Proposition 6.3 and Lemma 6.7, we can construct up-down queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$ in $\mathbf{Q}_{\mathtt{tree}}(\mathcal{F} - \{\cap\})$ with $e_1' \equiv_{\text{path}} \mathcal{Q}_1$ and $e_2' \equiv_{\text{path}} \mathcal{Q}_2$. By Proposition 6.10, we can construct up-down query $\mathcal{Q}$ in $\mathbf{Q}_{\mathtt{tree}}(\mathcal{F} - \{\cap\})$ such that, for every tree $\mathcal{T}$, $[\![\mathcal{Q}]\!]_{\mathcal{T}} = [\![\mathcal{Q}_1]\!]_{\mathcal{T}} \cap [\![\mathcal{Q}_2]\!]_{\mathcal{T}}$. Finally, by Proposition 6.8, we can construct a $\cup$-free expression e' in $\mathcal{N}(\mathcal{F} - \{\cap\})$ such that $e' \equiv_{\text{path}} \mathcal{Q}$. By the above construction, we conclude $e \equiv_{\text{path}} e'$. □

Theorem 6.11 yields the redundancy of intersection for local queries:

Corollary 6.12. *Let* $\{^{-1}, \pi\} \subseteq \mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}, \cap\}$. *On labeled trees, we have* $\mathcal{N}(\mathcal{F}) \equiv_{\text{path}} \mathcal{N}(\mathcal{F} - \{\cap\})$.

6.3. The Boolean equivalence of projection and converse

Fletcher et al. [18, 19, 29–31] already showed that, on graphs, $\mathcal{N}(^{-1}) \preceq_{\text{bool}} \mathcal{N}(\pi)$ (Proposition 8.4 in Section 8). On labeled trees, this result can be strengthened by showing that projections and the converse operator have equivalent expressive power in Boolean queries. To prove this, we need an intermediate result on $\mathbf{Q}_{\mathtt{tree}}()$:

Proposition 6.13. *For every condition tree query* $\mathcal{Q}$ *in* $\mathbf{Q}_{\mathtt{tree}}()$, *there exists a* $\cup$*-free expression in* $\mathcal{N}(^{-1})$ *such that* $\mathcal{Q} \equiv_{\text{bool}} e$.

Proof. Let $\mathcal{Q} = ((\mathcal{V}, \Sigma, \mathbf{E}), C, \mathsf{s}, \mathsf{t}, \gamma)$ and let r be the root node of tree $(\mathcal{V}, \Sigma, \mathbf{E})$. Construct $\mathcal{Q}_r = ((\mathcal{V}, \Sigma, \mathbf{E}), C, r, r, \gamma)$. Observe that $\mathcal{Q}_r$ in $\mathbf{Q}_{\mathtt{tree}}()$ and we have, for every tree $\mathcal{T}$, $[\![\mathcal{Q}]\!]_{\mathcal{T}} \neq \emptyset$ if and only if $[\![\mathcal{Q}_r]\!]_{\mathcal{T}} \neq \emptyset$. Hence, we have $\mathcal{Q} \equiv_{\text{bool}} \mathcal{Q}_r$.

Before we translate $\mathcal{Q}_r$ to an expression in $\mathcal{N}(^{-1})$, we reduce $\mathcal{Q}_r$ to a single node. We do so by a leaf-prune construction based on the construction used in the proof of Lemma 6.7. Let $n \in \mathcal{V} - \{r\}$ be a leaf node and let $(m, n) \in \mathbf{E}(\ell)$ be the edge that connects n to its single parent m. As $\mathsf{conditions}(n)$ is a node expression, we have $\pi_1[\ell \circ \mathsf{conditions}(n)] \equiv_{\text{path}} \ell \circ \mathsf{conditions}(n) \circ [\ell]^{-1}$. Hence, if $\mathsf{conditions}(n)$ in $\mathcal{N}(^{-1})$, then also $\ell \circ \mathsf{conditions}(n) \circ [\ell]^{-1}$ in $\mathcal{N}(^{-1})$. Remove n from the tree and add the node expression $\ell \circ \mathsf{conditions}(n) \circ [\ell]^{-1}$ to C, $\gamma(m)$, and let $\mathcal{Q}''$ be the result. By construction, we have $\mathcal{Q}''$ in $\mathbf{Q}_{\mathtt{tree}}(^{-1})$ and $\mathcal{Q}_r \equiv_{\text{path}} \mathcal{Q}''$.

Let $\mathcal{Q}'$ be a condition tree query in $\mathbf{Q}_{\mathtt{tree}}(^{-1})$ obtained by repeating the above leaf-prune operations on $\mathcal{Q}_r$ until only the root node r' remains. By construction, we have $\mathcal{Q}_r \equiv_{\text{path}} \mathcal{Q}' \equiv_{\text{path}} \mathsf{conditions}(r')$. As $\mathcal{Q}'$ in $\mathbf{Q}_{\mathtt{tree}}(^{-1})$, we have $\mathsf{conditions}(r')$ in $\mathcal{N}(^{-1})$ and we conclude $\mathcal{Q} \equiv_{\text{bool}} \mathsf{conditions}(r')$. □

Next, we observe that the class of condition tree queries $\mathbf{Q}_{\mathtt{tree}}()$, which consists of all condition-free condition tree queries, is equivalent to the tree queries of Wu et al. [14, Theorem 4.1]. Hence, we have

Lemma 6.14 (Wu et al.)**.** *For every* $\cup$*-free expression* e *in* $\mathcal{N}(\pi)$, *there exists a condition tree query* $\mathcal{Q}$ *in* $\mathbf{Q}_{\mathtt{tree}}()$ *with* $e \equiv_{\text{path}} \mathcal{Q}$.

Combining Proposition 6.13 and Lemma 6.14 yields the equivalence of projection and converse for local queries:

Corollary 6.15. *On labeled trees, we have* $\mathcal{N}(\pi) \equiv_{\text{bool}} \mathcal{N}(^{-1})$.

7. Adding the Kleene-star

In many practical graph query languages based on the relation algebra, such as the RPQs, 2RPQs, and nested RPQs, a form of recursion is added. Usually,

this is in the form of the *Kleene-star* operator: if e is an expression, then so is $[e]^*$ and the semantics of evaluation is defined by

$$[\![[e]^*]\!]_{\mathcal{G}} = \bigcup_{i \geq 0} [\![e^i]\!]_{\mathcal{G}}.$$

In the following, we extend the notation $\mathcal{N}(\mathcal{F})$ to $\mathcal{F} \subseteq \{\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap, -, {}^*\}$.

Adding the Kleene-star to any of the fragments of the relation algebra we consider can have an impact on the expressive power. Several facets of this have already been successfully studied in the literature by Benedikt et al. [11], Fletcher et al. [18, 19, 30], and Hellings et al. [12, 13] (see also Section 8). Furthermore, it is well-known that first-order logic cannot express that a chain has even length and cannot express the transitive closure [13, 23]. Hence, we have:

Lemma 7.1.

1. *Already on labeled chains, we have* $\mathcal{N}({}^*) \not\preceq_{\mathrm{bool}} \mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap, -)$.
2. *Already on unlabeled chains, we have* $\mathcal{N}({}^*) \not\preceq_{\mathrm{path}} \mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap, -)$.
3. *Already on unlabeled chains, we have* $\mathcal{N}(\overline{\pi}, {}^*) \not\preceq_{\mathrm{bool}} \mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap, -)$.

Additionally, the following collapses have been proven [12, 13, 29, 30]:

Lemma 7.2.

1. *On unlabeled graphs, we have* $\mathcal{N}(\mathcal{F} \cup \{{}^*\}) \preceq_{\mathrm{bool}} \mathcal{N}(\mathcal{F})$ *if* $\mathcal{F} \subseteq \{\mathrm{di}, \pi\}$.
2. *On unlabeled trees, we have* $\mathcal{N}({}^{-1}, \pi, \cap, {}^*) \preceq_{\mathrm{bool}} \mathcal{N}()$.

Next, we complete the picture. First, we show a final case in which Kleene-star does add expressive power to Boolean queries.

Proposition 7.3. *Let* $\mathcal{F} \subseteq \{\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap, -\}$. *Already on unlabeled trees, we have* $\mathcal{N}(\mathrm{di}, \cap, {}^*) \not\preceq_{\mathrm{bool}} \mathcal{N}(\mathcal{F})$.

Proof. As first-order logic cannot express that a chain (or any path between a pair of distinguishable nodes) has even length, we conclude that no expression in $\mathcal{N}(\mathcal{F})$ is Boolean-equivalent to $X \circ [\mathcal{E} \circ \mathcal{E}]^* \circ X$, in which $X = (\mathcal{E} \circ \mathrm{di}) \cap \mathcal{E}$ yields those edges from parent nodes to child nodes in which the parent also has other children. □

The above does not cover Boolean queries on unlabeled trees in $\mathcal{N}(\mathcal{F} \cup \{{}^*\})$, $\mathcal{F} \subseteq \{\mathrm{di}, {}^{-1}, \pi\}$. For these cases, we will show that we have $\mathcal{N}(\mathcal{F} \cup \{{}^*\}) \preceq_{\mathrm{bool}} \mathcal{N}(\mathcal{F})$. To do so, we first use subtree-reductions to obtain limits on the size of trees of a given depth:

Lemma 7.4. *Given a finite set of edge labels* Σ *and constants* $d, k \geq 0$.

1. *There exists only a finite number of trees* $\mathcal{T}$ *with* $\mathrm{depth}(\mathcal{T}) \leq d$ *that are not* k*-subtree-reducible.*
2. *There exists a constant* $w \geq 0$ *such that, for every tree* $\mathcal{T} = (\mathcal{V}, \Sigma, \mathbf{E})$ *with* $\mathrm{depth}(\mathcal{T}) \leq d$ *that is not* k*-subtree-reducible, we have* $|\mathcal{V}| \leq w$.

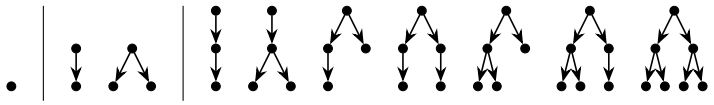

Figure 17: All non-2-subtree-reducible unlabeled trees of depth up-to-2.

Proof. By induction on the depth of trees, we show that every tree of depth d that is not k-subtree-reducible is isomorphic to a subtree of a single maximum-sized tree of depth d that is not k-subtree-reducible. The base case is $d = 0$, in which case we only have the tree with a single node. Now assume we have a set S_i of all trees of depth up-to-i, $i \geq 0$, and that these trees satisfy the Lemma. The biggest tree of depth $i + 1$ we can construct consists of a root node that has, per edge label $\ell \in \Sigma$ and per tree $\mathcal{T}$ in S_i, k outgoing edges labeled ℓ to children that are roots of copies of $\mathcal{T}$. All other trees of depth $i + 1$ that are not k-subtree-reducible are equivalent to a tree that can be constructed by taking a subset of the nodes and edges in this tree. □

Lemma 7.4 does not provide explicit upper bounds on the number of trees and the size of these trees. From the construction used in the proof of the lemma, it already follows that the upper bound grows at least exponentially fast with d, however.

Example 7.5. Figure 17 shows all unlabeled trees of depth up-to-2 that are not 2-subtree-reducible.

Next, we use Lemma 7.4 to prove the claimed collapse:

Theorem 7.6. *Let $\mathcal{F} \subseteq \{\mathrm{di}, ^{-1}, \pi\}$. On unlabeled trees, we have $\mathcal{N}(\mathcal{F} \cup \{^*\}) \preceq_{\mathrm{bool}} \mathcal{N}(\mathcal{F})$.*

Proof. Let e be an expression in $\mathcal{N}(\mathcal{F} \cup \{^*\})$ such that $e \not\equiv_{\mathrm{path}} \emptyset$. We perform a case distinction based on the depth of trees $\mathcal{T}$ on which e is evaluated.

Using a result from Hellings et al. [13, Lemma 3.10], there exists $k \geq 0$ such that $e \equiv_{\mathrm{bool}} \mathcal{E}^k$ on unlabeled chains. Let $\mathcal{T}$ be a tree with $\mathrm{depth}(\mathcal{T}) \geq k$ and let $\mathcal{C}$ be a chain with $\mathrm{depth}(\mathcal{C}) = k$. As $\mathrm{depth}(\mathcal{T}) \geq k$, we can map the chain $\mathcal{C}$ to a path from the root of $\mathcal{T}$ to a node in $\mathcal{T}$ with distance k. Using this mapping, we can apply another result of Hellings et al. [13, Lemma 3.9] to conclude that we have both $[\![\mathcal{E}^k]\!]_{\mathcal{T}} \neq \emptyset$ and $[\![e]\!]_{\mathcal{T}} \neq \emptyset$ whenever $\mathrm{depth}(\mathcal{T}) \geq k$.

Next, let $\mathcal{T}$ be a tree with $\mathrm{depth}(\mathcal{T}) < k$. Note that $[\![\mathcal{E}^k]\!]_{\mathcal{T}} = \emptyset$. Hence, it remains to find an expression e' in $\mathcal{N}(\mathcal{F})$ such that, for every tree $\mathcal{T}'$ with $\mathrm{depth}(\mathcal{T}') < k$, we have $[\![e]\!]_{\mathcal{T}'} \neq \emptyset$ if and only if $[\![e']\!]_{\mathcal{T}'} \neq \emptyset$. By Lemma 7.4(2), there exists an upper bound $w \geq 0$ for the size (in terms of the number of nodes) of any tree of depth up-to-k that is not 2-subtree-reducible. Construct e' from e by replacing every subexpression of the form $[g]^*$ by $\bigcup_{0 \leq i \leq w} g^i$. Let $\mathcal{T}'$ be a tree with $\mathrm{depth}(\mathcal{T}') < k$. We distinguish two cases:

1. *$\mathcal{T}'$ has at most w nodes.* In this case, $[\![e]\!]_{\mathcal{T}} = [\![e']\!]_{\mathcal{T}}$ by the semantics of the Kleene-star.

Table 1: Path equivalence relationships between language fragments of $\mathfrak{X}^{\uparrow}_{r,[\,]}$ and relation algebra fragments.

Benedikt et al.	Fragment of the relation algebra
$\mathfrak{X}^{\uparrow}_{[\,]}$	$\mathcal{N}(^{-1}, \pi_1)$
$\mathfrak{X}^{\uparrow}$	$\mathcal{N}(^{-1})$
$\mathfrak{X}_{[\,]}$	$\mathcal{N}(\pi_1)$
$\mathfrak{X}$	$\mathcal{N}()$

2. *$\mathcal{T}'$ has more than w nodes.* By Lemma 7.4(2), we can perform a sequence of 2-subtree-reduce steps on $\mathcal{T}'$ until we end up with a tree $\mathcal{T}''$ with at most w nodes. By the semantics of the Kleene-star, we have $[\![e]\!]_{\mathcal{T}''} = [\![e']\!]_{\mathcal{T}''}$. By Proposition 3.5(2), we have $[\![e']\!]_{\mathcal{T}''} \neq \emptyset$ if and only if $[\![e']\!]_{\mathcal{T}'} \neq \emptyset$. We conclude $[\![e']\!]_{\mathcal{T}'} \neq \emptyset$ if and only if $[\![e]\!]_{\mathcal{T}'} \neq \emptyset$.

We conclude $\mathcal{E}^k \cup e' \equiv_{\text{bool}} e$ and, by construction, $\mathcal{E}^k \cup e'$ is in $\mathcal{N}(\mathcal{F})$. □

8. Related work and results from the literature

The XPath query language for XML data has several formalizations. The formalization of Benedikt et al. [11] includes a study of the relative expressive power of fragments of $\mathfrak{X}^{\uparrow}_{r,[\,]}$ on node-labeled trees. The fragments of $\mathfrak{X}^{\uparrow}_{r,[\,]}$ are closely related to fragments of $\mathcal{N}(^{-1}, \pi, ^*)$. In Table 1, we provide the mapping between the query languages studied by Benedikt et al. and the query languages we study.

Additionally, the study of Benedikt et al. also considers fragments obtained by adding a weak form of the Kleene-star operator to the languages: the descendant axis $[\mathcal{E}]^*$ (and the ancestor axis $[[\mathcal{E}]^{-1}]^*$ if $^{-1}$ is included in the fragment). Without much effort, we can show that the results of Benedikt et al. on node-labeled trees also apply to the edge-labeled trees we study.

Proposition 8.1 (Benedikt et al. [11, Proposition 2.1])**.** *Let $\mathcal{F}_1, \mathcal{F}_2 \subseteq \{^{-1}, \pi, ^*\}$. Already on labeled trees, we have $\mathcal{N}(\mathcal{F}_1) \preceq_{\text{path}} \mathcal{N}(\mathcal{F}_2)$ if and only if $\mathcal{F}_1 \subseteq \mathcal{F}_2$.*

Wu et al. [14] studies the language Path$^+$ and its fragments on node-labeled trees. These languages are all path-equivalent to $\cup$-free fragments of the relation algebra. In Table 2, we provide the mapping between the query languages studied in Wu et al. and the query languages we study.

Among other things, Wu et al. showed that the query languages Path$^+$, Path$^+(\cap)$, and Path$^+(\pi_1, \pi_2)$ are path-equivalent on labeled trees. These results translate to our setting.

Proposition 8.2 (Wu et al. [14, Theorem 4.1])**.** *On labeled trees, every $\cup$-free expression in $\mathcal{N}(^{-1}, \pi, \cap)$ is path-equivalent to a $\cup$-free expression in $\mathcal{N}(^{-1}, \pi)$ and in $\mathcal{N}(^{-1}, \cap)$.*

Table 2: Path-subsumption relationship between language fragments of Path^+ and relation algebra fragments.

Wu et al.	Fragment of the relation algebra
Path^+	$\mathcal{N}(^{-1}, \pi, \cap)$
$\text{Path}^+(\cap)$	$\mathcal{N}(^{-1}, \cap)$
$\text{Path}^+(\pi_1, \pi_2)$	$\mathcal{N}(^{-1}, \pi)$
$\text{DPath}^+(\pi_1)$	$\mathcal{N}(\pi_1)$

Benedikt et al. and Wu et al. only studied fragments that are monotone and 1-subtree-reducible. From our results, we conclude that these fragments have only very limited expressive power, especially with respect to detecting branching structures in trees. Our work extends the results for these limited fragments to the more expressive full relation algebra, to Boolean queries, to unlabeled trees, and to chains.

Finally, Hellings et al. [12, 13] studied the relative expressiveness of the fragments of $\mathcal{N}(\pi, \overline{\pi}, \cap, -, ^*)$,[7] and obtained the following results which are relevant for this study:

Proposition 8.3 ([13, Theorem 3.1])**.** *Let $\mathcal{F}_1, \mathcal{F}_2 \subseteq \{\pi, \overline{\pi}, \cap, -, ^*\}$.*

1. *On labeled and unlabeled trees, we have $\mathcal{N}(\mathcal{F}_1) \preceq_{\text{path}} \mathcal{N}(\mathcal{F}_2)$ if and only if $(\underline{\mathcal{F}_1} - \{\cap, -\}) \subseteq (\underline{\mathcal{F}_2} - \{\cap, -\})$.*
2. *On labeled trees, we have $\mathcal{N}(\mathcal{F}_1) \preceq_{\text{bool}} \mathcal{N}(\mathcal{F}_2)$ if and only if $(\underline{\mathcal{F}_1} - \{\cap, -\}) \subseteq (\underline{\mathcal{F}_2} - \{\cap, -\})$.*

Additionally, the following results are obtained:

4. *On unlabeled trees, $\mathcal{N}(^{-1}) \not\preceq_{\text{path}} \mathcal{N}(\pi, \overline{\pi}, \cap, -, ^*)$ and $\mathcal{N}(\text{di}) \not\preceq_{\text{path}} \mathcal{N}(\pi, \overline{\pi}, \cap, -, ^*)$.*
5. *On unlabeled trees, $\mathcal{N}(^{-1}, \pi, \cap, ^*) \preceq_{\text{bool}} \mathcal{N}()$.*
6. *On unlabeled trees, $\mathcal{N}(\overline{\pi}) \not\preceq_{\text{bool}} \mathcal{N}(\text{di}, ^{-1}, \pi, \cap, ^*)$, and $\mathcal{N}(\overline{\pi}, ^*) \not\preceq_{\text{bool}} \mathcal{N}(\overline{\pi})$.*

On graphs, the relative expressive power of fragments of the relation algebra has been studied in great detail by Fletcher et al. [18, 19, 29–31]. A close inspection of the individual separation results of Fletcher et al. reveals that none of these separation results directly apply to trees or chains, as the proofs of these separation results rely on non-tree graph structures. Consequently, our work improves on these results significantly by showing that many separations already hold on the much simpler tree and chain data models. Fletcher et al. also proved a collapse result, that automatically also holds on trees:

Proposition 8.4 (Fletcher et al. [19, Proposition 4.2])**.** *Let $\mathcal{F} \subseteq \{\text{di}, ^{-1}, \pi, \overline{\pi}\}$. On labeled and unlabeled graphs, we have $\mathcal{N}(\mathcal{F} \cup \{^{-1}\}) \preceq_{\text{bool}} \mathcal{N}(\mathcal{F} \cup \{\pi\})$ if $\mathcal{F} \subseteq \{\text{di}, ^{-1}, \pi, \overline{\pi}\}$.*

[7] These are generally referred to as the *downward* fragments of the relation algebra.

Additionally, Fletcher et al. also studied the Kleene-star, for which the following collapse results is provided:

Proposition 8.5 (Fletcher et al. [30, Theorem 1]). *Let $\mathcal{F} \subseteq \{\mathrm{di}, {}^{-1}, \pi, \overline{\pi}\}$. On unlabeled graphs, we have $\mathcal{N}(\mathcal{F} \cup \{^*\}) \preceq_{\mathrm{bool}} \mathcal{N}(\mathcal{F})$ if $\mathcal{F} \subseteq \{\mathrm{di}, \pi\}$.*

Theorem 7.6 already showed that, on trees, this collapse result can be strengthened significantly.

Several other well-known expressiveness results are known in the context of Conditional XPath and Navigational XPath [6, 8, 32], which are strongly related to the relation algebra. Unfortunately, these results are proved with respect to the sibling-ordered tree data model, and do not apply to our unordered tree data model.

9. Conclusion and future work

In this paper, we settled the relative expressive power of queries in fragments of the relation algebra when used to query trees. A summary of our results can be found in Figure 18. To compensate for the limited flexibility of the tree data model compared to the graph data model, we needed to develop several new techniques to make this study feasible. For the local fragments, i.e., fragments of $\mathcal{N}({}^{-1}, \pi, \overline{\pi}, \cap, -)$, we provided a complete characterization of their relative expressive power, and with respect to the non-local fragments, only one challenging problem remains open:

Problem 9.1. *Let $\{\mathrm{di}, \overline{\pi}, \cap\} \subseteq \mathcal{F} \subseteq \{\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap\}$, let $\mathbb{G}$ be the class of labeled trees, unlabeled trees, or labeled chains, and let $sem \in \{\mathrm{bool}, \mathrm{path}\}$. Do we have $\mathcal{N}(\mathcal{F} \cup \{-\}) \preceq_{sem,\mathbb{G}} \mathcal{N}(\mathcal{F})$ or not?*

The difficulties in solving this open problem are manifold. For example, consider the language $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap)$. In this fairly rich query language, there are many expressions for which one can express the complement. E.g., for $k > 0$, we have $\overline{\mathcal{E}^{-k}} \equiv_{\mathrm{path}} (\mathcal{E}^{-k} \circ \mathrm{di}) \cup (\overline{\pi}_1[\mathcal{E}^{-k}] \circ (\mathrm{id} \cup \mathrm{di}))$. Unfortunately, we have not been able yet to express the complement for every expression in $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap)$, and we have also not been able to prove that expressing the complement of some expressions in $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap)$ is impossible in $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap)$.

On unlabeled chains, we already established that $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap, -) \preceq_{\mathrm{bool}} \mathcal{N}(\mathrm{di}, \overline{\pi}, \cap)$, as every first-order query is Boolean-equivalent to an expression in $\mathcal{N}(\overline{\pi})$ [28, Section 3.9]. At the same time, we already know that $\mathcal{N}(\mathcal{F} \cup \{-\}) \not\preceq_{\mathrm{bool}} \mathcal{N}(\mathcal{F})$, $\mathcal{F} \subseteq \{\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap\}$, on unlabeled graphs [18, 19], a result that can already be obtained using brute-force methods. Such brute-force results cannot be extended to simpler classes of graphs, however. Indeed, we can prove that *strong Boolean separations*, which can be proven using brute-force methods, do not exist in our open cases. In specific, we have the following

Lemma 9.2 (Hellings [28, Chapter 6.2]). *Let $\mathcal{T}$ be a tree that is not 3-subtree-reducible. There exist expressions e_1 in $\mathcal{N}(\mathrm{di}, {}^{-1}, \overline{\pi}, \cap)$ and e_2 in $\mathcal{N}({}^{-1}, -)$ such that, for every $\mathcal{T}'$ that is not 3-subtree-reducible, $[\![e_1]\!]_{\mathcal{T}'} \neq \emptyset$ and $[\![e_2]\!]_{\mathcal{T}'} \neq \emptyset$ if and only if $\mathcal{T}$ is isomorphic to $\mathcal{T}'$.*

From the above, it follows that a possible separation between $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap)$ and $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap, -)$ cannot be established on a single pair of trees, ruling out the applicability of brute-force techniques and many techniques developed in our work. Hence, we definitely face a challenging open problem, which we hope to solve in the future.

References

[1] J. Hellings, Y. Wu, M. Gyssens, D. V. Gucht, The power of Tarski's relation algebra on trees, in: Proceedings of the 10th International Symposium on Foundations of Information and Knowledge Systems, Springer International Publishing, 2018, pp. 244–264. doi:10.1007/978-3-319-90050-6_14.

[2] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, J. Cowan, Extensible markup language (XML) 1.1 (second edition), W3C recommendation, W3C (2006).
URL http://www.w3.org/TR/2006/REC-xml11-20060816

[3] Ecma International, The JSON data interchange syntax, 2nd edition (2017).
URL http://www.ecma-international.org/publications/standards/Ecma-404.htm

[4] D. C. Tsichritzis, F. H. Lochovsky, Hierarchical data-base management: A survey, ACM Computing Surveys 8 (1) (1976) 105–123. doi:10.1145/356662.356667.

[5] M. Benedikt, C. Koch, XPath leashed, ACM Computing Surveys 41 (1) (2009) 3:1–3:54. doi:10.1145/1456650.1456653.

[6] B. ten Cate, The expressivity of XPath with transitive closure, in: Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, 2006, pp. 328–337. doi:10.1145/1142351.1142398.

[7] J. Clark, S. DeRose, XML path language (XPath) version 1.0, W3C recommendation, W3C (1999).
URL http://www.w3.org/TR/1999/REC-xpath-19991116

[8] M. Marx, Conditional XPath, ACM Transactions on Database Systems 30 (4) (2005) 929–959. doi:10.1145/1114244.1114247.

[9] J. Hidders, J. Paredaens, J. Van den Bussche, J-Logic: Logical foundations for JSON querying, in: Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, ACM, 2017, pp. 137–149. doi:10.1145/3034786.3056106.

[10] A. Tarski, On the calculus of relations, The Journal of Symbolic Logic 6 (3) (1941) 73–89. doi:10.2307/2268577.

[11] M. Benedikt, W. Fan, G. Kuper, Structural properties of XPath fragments, Theoretical Computer Science 336 (1) (2005) 3–31. doi:10.1016/j.tcs.2004.10.030.

[12] J. Hellings, M. Gyssens, Y. Wu, D. Van Gucht, J. Van den Bussche, S. Vansummeren, G. H. L. Fletcher, Relative expressive power of downward fragments of navigational query languages on trees and chains, in: Proceedings of the 15th Symposium on Database Programming Languages, ACM, 2015, pp. 59–68. doi:10.1145/2815072.2815081.

[13] J. Hellings, M. Gyssens, Y. Wu, D. V. Gucht, J. V. den Bussche, S. Vansummeren, G. H. Fletcher, Comparing the expressiveness of downward fragments of the relation algebra with transitive closure on trees, Information Systems 89 (2020). doi:10.1016/j.is.2019.101467.

[14] Y. Wu, D. Van Gucht, M. Gyssens, J. Paredaens, A study of a positive fragment of path queries: Expressiveness, normal form and minimization, The Computer Journal 54 (7) (2011) 1091–1118. doi:10.1093/comjnl/bxq055.

[15] I. F. Cruz, A. O. Mendelzon, P. T. Wood, A graphical query language supporting recursion, in: Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data, ACM, 1987, pp. 323–330. doi:10.1145/38713.38749.

[16] P. Barceló, Querying graph databases, in: Proceedings of the 32nd Symposium on Principles of Database Systems, ACM, 2013, pp. 175–188. doi:10.1145/2463664.2465216.

[17] P. Barceló, J. Pérez, J. L. Reutter, Relative expressiveness of nested regular expressions, in: Proceedings of the 6th Alberto Mendelzon International Workshop on Foundations of Data Management, CEUR Workshop Proceedings, 2012, pp. 180–195.

[18] 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 of the 14th International Conference on Database Theory, ACM, 2011, pp. 197–207. doi:10.1145/1938551.1938578.

[19] 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 querying on graphs, Information Sciences 298 (2015) 390–406. doi:10.1016/j.ins.2014.11.031.

[20] S. Givant, The calculus of relations as a foundation for mathematics, Journal of Automated Reasoning 37 (4) (2006) 277–322. doi:10.1007/s10817-006-9062-x.

[21] M. P. Consens, A. O. Mendelzon, GraphLog: A visual formalism for real life recursion, in: Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM, 1990, pp. 404–416. doi:10.1145/298514.298591.

[22] D. Calvanese, G. D. Giacomo, M. Lenzerini, M. Y. Vardi, Containment of conjunctive regular path queries with inverse, in: Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann Publishers Inc., 2000, pp. 176–185.

[23] L. Libkin, Elements of Finite Model Theory, Springer Berlin Heidelberg, 2004.

[24] M. Grohe, Finite variable logics in descriptive complexity theory, The Bulletin of Symbolic Logic 4 (1998) 345–398. doi:10.2307/420954.

[25] E. Grädel, M. Otto, On logics with two variables, Theoretical Computer Science 224 (1–2) (1999) 73–113. doi:10.1016/S0304-3975(98)00308-9.

[26] R. Kaushik, P. Bohannon, J. F. Naughton, H. F. Korth, Covering indexes for branching path queries, in: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, ACM, 2002, pp. 133–144. doi:10.1145/564691.564707.

[27] J. Hellings, C. L. Pilachowski, D. Van Gucht, M. Gyssens, Y. Wu, From relation algebra to semi-join algebra: An approach for graph query optimization, in: Proceedings of the 16th International Symposium on Database Programming Languages, ACM, 2017, pp. 5:1–5:10. doi:10.1145/3122831.3122833.

[28] J. Hellings, On tarski's relation algebra: querying trees and chains and the semi-join algebra, Ph.D. thesis, Hasselt University and transnational University of Limburg (2018).

[29] 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 boolean expressiveness of navigational query languages on graphs, in: Proceedings of the 7th International Symposium on Foundations of Information and Knowledge Systems, Vol. 7153, Springer Berlin Heidelberg, 2012, pp. 124–143.

[30] 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, Annals of Mathematics and Artificial Intelligence 73 (1-2) (2015) 167–203. doi:10.1007/s10472-013-9346-x.

[31] D. Surinx, 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 using transitive closure, Logic Journal of the IGPL 23 (5) (2015) 759–788. doi:10.1093/jigpal/jzv028.

[32] M. Marx, M. de Rijke, Semantic characterizations of navigational XPath, SIGMOD Record 34 (2) (2005) 41–46. doi:10.1145/1083784.1083792.

Unlabeled trees

	Boolean semantics							Path semantics						
	di	$^{-1}$	π	$\overline{\pi}$	$\cap$	$-$	*	di	$^{-1}$	π	$\overline{\pi}$	$\cap$	$-$	*
$\mathcal{N}()$	3.7 ✗	8.3 ✓	8.3 ✓	8.3 ✗	8.3 ✓	8.3 ✓	8.3 ✓	8.3 ✗	8.3 ✗	8.3 ✗	8.3 ✗	8.3 ✓	8.3 ✓	8.3 ✗
$\mathcal{N}(\cap)$	3.7 ✗	8.3 ✓	8.3 ✓	8.3 ✗		8.3 ✓	8.3 ✓	8.3 ✗	8.3 ✗	8.3 ✗	8.3 ✗		8.3 ✓	8.3 ✗
$\mathcal{N}(\cap,-)$	8.3 ✗	8.3 ✗	8.3 ✗	8.3 ✗			8.3 ✓	8.3 ✗	8.3 ✗	8.3 ✗	8.3 ✗			8.3 ✗
$\mathcal{N}(\pi)$	3.7 ✗	8.4 ✓		8.3 ✗	8.3 ✓	8.3 ✗	8.5 ✓	8.3 ✗	8.3 ✗		8.3 ✗	8.3 ✓	8.3 ✗	8.3 ✗
$\mathcal{N}(\pi,\cap)$	3.7 ✗	8.2 ✓		8.3 ✗		8.3 ✗	8.3 ✓	8.3 ✗	8.3 ✗		8.3 ✗		8.3 ✗	8.3 ✗
$\mathcal{N}(\pi,\overline{\pi})$	3.7 ✗	8.4 ✓			8.3 ✓	8.3 ✓	8.3 ✗	8.3 ✗	8.3 ✗			8.3 ✓	8.3 ✓	8.3 ✗
$\mathcal{N}(\pi,\overline{\pi},\cap)$	3.7 ✗	6.12 ✓				8.3 ✓	8.3 ✗	8.3 ✗	8.3 ✗				8.3 ✓	8.3 ✗
$\mathcal{N}(\pi,\overline{\pi},\cap,-)$	3.7 ✗	3.7 ✗					8.3 ✗	8.3 ✗	8.3 ✗					8.3 ✗
$\mathcal{N}(^{-1})$	3.7 ✗		8.3 ✓	8.3 ✗	8.3 ✓	8.3 ✗	8.3 ✓	3.7 ✗		5.1 ✗	8.3 ✗	5.1 ✗	8.3 ✗	8.3 ✗
$\mathcal{N}(^{-1},\pi)$	3.7 ✗			8.3 ✗	8.2 ✓	8.3 ✗	8.3 ✓	3.7 ✗			8.3 ✗	8.2 ✓	8.3 ✗	8.3 ✗
$\mathcal{N}(^{-1},\pi,\cap)$	3.7 ✗			8.3 ✗		8.3 ✗	8.3 ✓	3.7 ✗			8.3 ✗		8.3 ✗	8.3 ✗
$\mathcal{N}(^{-1},\pi,\overline{\pi})$	3.7 ✗				6.12 ✓	3.7 ✗	8.3 ✗	3.7 ✗				6.12 ✓	3.7 ✗	8.3 ✗
$\mathcal{N}(^{-1},\pi,\overline{\pi},\cap)$	3.7 ✗					3.7 ✗	8.3 ✗	3.7 ✗					3.7 ✗	8.3 ✗
$\mathcal{N}(^{-1},\pi,\overline{\pi},\cap,-)$	4.5 ✗						8.3 ✗	4.5 ✗						8.3 ✗
$\mathcal{N}(\text{di})$		5.2 ✗	5.2 ✗	8.3 ✗	3.7 ✗	8.3 ✗	8.5 ✓		5.2 ✗	5.2 ✗	8.3 ✗	3.7 ✗	8.3 ✗	8.3 ✗
$\mathcal{N}(\text{di},\pi)$		8.4 ✓		8.3 ✗	3.7 ✗	8.3 ✗	8.5 ✓		4.10 ✗		8.3 ✗	3.7 ✗	8.3 ✗	8.3 ✗
$\mathcal{N}(\text{di},\pi,\cap)$		5.2 ✗		8.3 ✗		8.3 ✗	7.3 ✗		5.2 ✗		8.3 ✗		8.3 ✗	8.3 ✗
$\mathcal{N}(\text{di},\pi,\overline{\pi})$		8.4 ✓			3.7 ✗	3.7 ✗	8.3 ✗		5.1 ✗			3.7 ✗	3.7 ✗	8.3 ✗
$\mathcal{N}(\text{di},\pi,\overline{\pi},\cap)$		5.2 ✗				?	8.3 ✗		5.2 ✗				?	8.3 ✗
$\mathcal{N}(\text{di},\pi,\overline{\pi},\cap,-)$		5.2 ✗					8.3 ✗		5.2 ✗					8.3 ✗
$\mathcal{N}(\text{di},^{-1})$			5.2 ✗	8.3 ✗	3.7 ✗	8.3 ✗	7.6 ✓			5.2 ✗	8.3 ✗	3.7 ✗	8.3 ✗	8.3 ✗
$\mathcal{N}(\text{di},^{-1},\pi)$				8.3 ✗	3.7 ✗	8.3 ✗	7.6 ✓				8.3 ✗	3.7 ✗	8.3 ✗	8.3 ✗
$\mathcal{N}(\text{di},^{-1},\pi,\cap)$				8.3 ✗		8.3 ✗	7.3 ✗				8.3 ✗		8.3 ✗	8.3 ✗
$\mathcal{N}(\text{di},^{-1},\pi,\overline{\pi})$					3.7 ✗	3.7 ✗	8.3 ✗					3.7 ✗	3.7 ✗	8.3 ✗
$\mathcal{N}(\text{di},^{-1},\pi,\overline{\pi},\cap)$						?	8.3 ✗						?	8.3 ✗
$\mathcal{N}(\text{di},^{-1},\pi,\overline{\pi},\cap,-)$							8.3 ✗							8.3 ✗

Labeled trees

	Boolean semantics							Path semantics						
	di	$^{-1}$	π	$\overline{\pi}$	$\cap$	$-$	*	di	$^{-1}$	π	$\overline{\pi}$	$\cap$	$-$	*
$\mathcal{N}()$	3.7 ✗	8.3 ✗	8.3 ✗	8.3 ✗	8.3 ✓	8.3 ✓	8.3 ✗	8.3 ✗	8.1 ✗	8.1 ✗	8.3 ✗	8.3 ✓	8.3 ✓	8.1 ✗
$\mathcal{N}(\cap)$	3.7 ✗	8.3 ✗	8.3 ✗	8.3 ✗		8.3 ✓	8.3 ✗	8.3 ✗	8.3 ✗	8.3 ✗	8.3 ✗		8.3 ✓	8.3 ✗
$\mathcal{N}(\cap,-)$	8.3 ✗	8.3 ✗	8.3 ✗	8.3 ✗			8.3 ✗	8.3 ✗	8.3 ✗	8.3 ✗	8.3 ✗			8.3 ✗
$\mathcal{N}(\pi)$	3.7 ✗	8.4 ✓		8.3 ✗	8.3 ✓	8.3 ✗	8.3 ✗	8.3 ✗	8.1 ✗		8.3 ✗	8.3 ✓	8.3 ✗	8.1 ✗
$\mathcal{N}(\pi,\cap)$	3.7 ✗	8.2 ✓		8.3 ✗		8.3 ✗	8.3 ✗	8.3 ✗	8.3 ✗		8.3 ✗		8.3 ✗	8.3 ✗
$\mathcal{N}(\pi,\overline{\pi})$	3.7 ✗	8.4 ✓			8.3 ✓	8.3 ✓	8.3 ✗	8.3 ✗	8.3 ✗			8.3 ✓	8.3 ✓	8.3 ✗
$\mathcal{N}(\pi,\overline{\pi},\cap)$	3.7 ✗	6.12 ✓				8.3 ✓	8.3 ✗	8.3 ✗	8.3 ✗				8.3 ✓	8.3 ✗
$\mathcal{N}(\pi,\overline{\pi},\cap,-)$	3.7 ✗	3.7 ✗					8.3 ✗	8.3 ✗	8.3 ✗					8.3 ✗
$\mathcal{N}(^{-1})$	3.7 ✗		6.15 ✓	8.3 ✗	6.15 ✓	8.3 ✗	8.3 ✗	3.7 ✗		8.1 ✗	8.3 ✗	5.1 ✗	8.3 ✗	8.1 ✗
$\mathcal{N}(^{-1},\pi)$	3.7 ✗			8.3 ✗	8.2 ✓	8.3 ✗	8.3 ✗	3.7 ✗			8.3 ✗	8.2 ✓	8.3 ✗	8.1 ✗
$\mathcal{N}(^{-1},\pi,\cap)$	3.7 ✗			8.3 ✗		8.3 ✗	8.3 ✗	3.7 ✗			8.3 ✗		8.3 ✗	8.3 ✗
$\mathcal{N}(^{-1},\pi,\overline{\pi})$	3.7 ✗				6.12 ✓	3.7 ✗	8.3 ✗	3.7 ✗				6.12 ✓	3.7 ✗	8.3 ✗
$\mathcal{N}(^{-1},\pi,\overline{\pi},\cap)$	3.7 ✗					3.7 ✗	8.3 ✗	3.7 ✗					3.7 ✗	8.3 ✗
$\mathcal{N}(^{-1},\pi,\overline{\pi},\cap,-)$	4.4 ✗						8.3 ✗	4.4 ✗						8.3 ✗
$\mathcal{N}(\text{di})$		5.2 ✗	5.2 ✗	8.3 ✗	3.7 ✗	8.3 ✗	8.3 ✗		5.2 ✗	5.2 ✗	8.3 ✗	3.7 ✗	8.3 ✗	8.3 ✗
$\mathcal{N}(\text{di},\pi)$		8.4 ✓		8.3 ✗	3.7 ✗	8.3 ✗	8.3 ✗		4.10 ✗		8.3 ✗	3.7 ✗	8.3 ✗	8.3 ✗
$\mathcal{N}(\text{di},\pi,\cap)$		5.2 ✗		8.3 ✗		8.3 ✗	7.3 ✗		5.2 ✗		8.3 ✗		8.3 ✗	8.3 ✗
$\mathcal{N}(\text{di},\pi,\overline{\pi})$		8.4 ✓			3.7 ✗	3.7 ✗	8.3 ✗		5.1 ✗			3.7 ✗	3.7 ✗	8.3 ✗
$\mathcal{N}(\text{di},\pi,\overline{\pi},\cap)$		5.2 ✗				?	8.3 ✗		5.2 ✗				?	8.3 ✗
$\mathcal{N}(\text{di},\pi,\overline{\pi},\cap,-)$		5.2 ✗					8.3 ✗		5.2 ✗					8.3 ✗
$\mathcal{N}(\text{di},^{-1})$			5.2 ✗	8.3 ✗	3.7 ✗	8.3 ✗	8.3 ✗			5.2 ✗	8.3 ✗	3.7 ✗	8.3 ✗	8.3 ✗
$\mathcal{N}(\text{di},^{-1},\pi)$				8.3 ✗	3.7 ✗	8.3 ✗	8.3 ✗				8.3 ✗	3.7 ✗	8.3 ✗	8.3 ✗
$\mathcal{N}(\text{di},^{-1},\pi,\cap)$				8.3 ✗		8.3 ✗	7.3 ✗				8.3 ✗		8.3 ✗	8.3 ✗
$\mathcal{N}(\text{di},^{-1},\pi,\overline{\pi})$					3.7 ✗	3.7 ✗	8.3 ✗					3.7 ✗	3.7 ✗	8.3 ✗
$\mathcal{N}(\text{di},^{-1},\pi,\overline{\pi},\cap)$						?	8.3 ✗						?	8.3 ✗
$\mathcal{N}(\text{di},^{-1},\pi,\overline{\pi},\cap,-)$							8.3 ✗							8.3 ✗

Figure 18: Index to the separation and collapse results discussed in this paper. Let $(\mathcal{N}(\mathcal{F}), op)$, $\mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}, \cap, -\}$, be a field in the "*sem* semantics" part of the table, $sem \in \{\text{bool}, \text{path}\}$. A check mark ✓ (green-colored cells) in the field $(\mathcal{N}(\mathcal{F}), op)$ means that $\mathcal{N}(\mathcal{F} \cup \{op\}) \preceq_{sem} \mathcal{N}(\mathcal{F})$. A cross ✗ (red-colored cells) in the field $(\mathcal{N}(\mathcal{F}), op)$ means that $\mathcal{N}(\mathcal{F} \cup \{op\}) \not\preceq_{sem} \mathcal{N}(\mathcal{F})$. Finally, a question mark **?** (yellow-colored cells) indicates an open problem. The cases in gray-colored cells indicate results from related work (Section 8). For brevity of the paper, some results are only obtained by combining the referenced result with a general collapse result such as Corollary 6.12.