The power of Tarski's relation algebra on trees*

Jelle Hellings[1], Yuqing Wu[2], Marc Gyssens[1], and Dirk Van Gucht[3]

[1] Hasselt University, Martelarenlaan 42, 3500, Hasselt, Belgium
[2] Pomona College, 185 E 6th St., Claremont, CA 91711, USA
[3] Indiana University, 150 S. Woodlawn Ave, Bloomington, IN 47405, 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. 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, intersections, and/or difference, both for path queries and Boolean queries. For path queries, 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, 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. One challenging problem remains open, however, for both path and Boolean queries: does adding difference yields more expressive power to fragments containing at least diversity, coprojections, and intersection?

1 Introduction

Trees can be used to model data that has a hierarchical or nested structure including taxonomies, organizational charts, documents, genealogies, and file and directory structures. It is therefore not surprising that tree data models have been continuously studied since the 1960s [5,9,25]. Modern query languages for querying tree data have a heavy reliance on navigating the tree structure. Prime examples of this are XPath [4,6,7,22] and the various JSON query languages [19]. At its core, this navigation can be captured by fragments of Tarski's relation algebra [24]. Consequently, tree querying based on fragments of the relation algebra has already been studied in great detail (e.g. [3,16,17,26]). 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.

In this work, we undertake such a comprehensive study by investigating the relative expressive power of fragments of the relation algebra with respect to both

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

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) [8]. 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 [1]. 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 [2]. 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. Finally, we study adding negation in the form of coprojections ($\overline{\pi}_1$ and $\overline{\pi}_2$) and difference ($-$). All the above notations that are at the basis of this study can be found in Section 2.

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 [10,11,12,24], this simplicity makes it much more difficult to establish separation results using strong brute-force methods. 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, 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. This in turn yields several separation results on trees:

1. *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 and trees. Other query languages support recognizing branching up to a certain degree, and we can classify these languages accordingly. To do so, we introduce a notion called *k-subtree reductions* 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, the query language $\mathcal{N}(^{-1}, \pi, \overline{\pi}, \cap)$ is 1-*subtree-reducible* and, consequently, can only recognize siblings if they are not structurally equivalent. Next, query languages $\mathcal{N}(\mathcal{F})$ with $\mathrm{di} \in \mathcal{F}$ and $\cap \notin \overline{\mathcal{F}}$ are 2-*subtree-reducible* and can, in very limited circumstances, distinguish up to two structurally equivalent children of a node. Finally the full relation algebra is 3-*subtree-reducible*, and query languages $\mathcal{N}(\mathcal{F})$ with $\{^{-1}, -\} \subseteq \mathcal{F}$ or $\{\mathrm{di}, \cap\} \subseteq \mathcal{F}$ can always distinguish between nodes that have one, two, or at least three structurally equivalent children.

2. *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 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. 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".

3. *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. Hence, we call these query languages *downward* [16,17]. We observe that these downward query languages are all 1-subtree-reducible, which puts an upper bound on their expressive power. Diversity and the converse operator are *non-downward* in nature. Based on this observation, it follows that languages with diversity or converse are not path-equivalent to downward query languages.

4. *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 $\mathcal{G}$, we have $[\![q]\!]_{\mathcal{G}} \subseteq [\![q]\!]_{\mathcal{G}'}$. One the one hand, one can show that the query language $\mathcal{N}(\mathrm{di}, ^{-1}, \pi, \cap)$ is monotone [16,17]. On the other hand, the query languages $\mathcal{N}(\mathcal{F})$ with $\overline{\pi} \in \overline{\mathcal{F}}$ 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. Such queries are not monotone and consequently not expressible in $\mathcal{N}(\mathrm{di}, ^{-1}, \pi, \cap)$.

In Figure 1, 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 12.

Some separation results are obtained through brute-force methods, which we will show 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.

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 claim this a very challenging open case, and we consider identifying it as the third major contribution of this paper. We discuss this open case in Section 8, in which we also discuss other directions for future research.

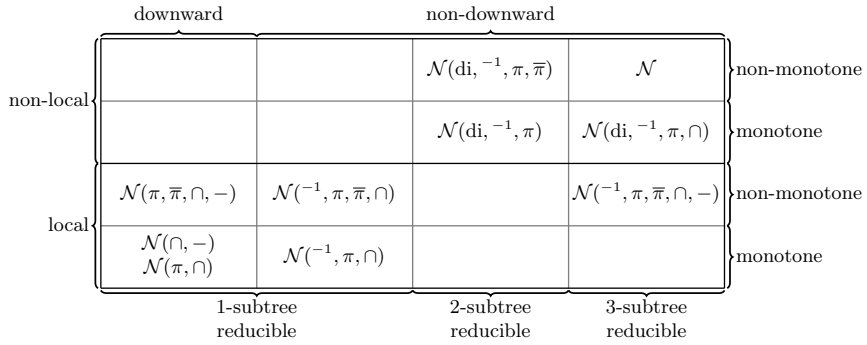

Fig. 1. Initial classification of the relative expressive power of fragments of the relation algebra with respect to path queries on labeled trees. In each box, the largest fragment(s) that satisfy the classification of that particular box are included. The more to the right and to the top a certain box is situated, the stronger the expressiveness of the corresponding fragment(s) become.

2 Preliminaries[4]

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 labels, and $\mathbf{E} : \Sigma \to 2^{\mathcal{V}\times\mathcal{V}}$ a function mapping labels to edge relations. We denote by $\mathcal{E}$ the union of all edge relations. If $|\Sigma| = 1$, $\mathcal{G}$ is *unlabeled*. A *tree* $\mathcal{T} = (\mathcal{V}, \Sigma, \mathbf{E})$ is a connected acyclic graph in which one node, the *root*, has no incoming edges, and all other nodes have one incoming edge. In an edge $(m, n) \in \mathcal{E}$, m is the *parent* of n, and n a *child* of m. A *chain* is a tree in which all nodes have at most one child.

In this paper, we limit our study to queries on trees and chains. A query q maps a tree to a set of node pairs. We write $[\![q]\!]_{\mathcal{T}}$ to denote the *evaluation* of q on tree $\mathcal{T}$. We can interpret a query q literally as a *path query*, or, alternatively, as a *Boolean query*, in which case `True` stands for $[\![q]\!]_{\mathcal{T}} \neq \emptyset$.

The syntax of a *relation algebra* expression is given by

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

where $\ell \in \Sigma$ and $j \in \{1, 2\}$. Its evaluation on a tree $\mathcal{T} = (\mathcal{V}, \Sigma, \mathbf{E})$ is defined by

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

[4] Our formalization of graphs, the relation algebra, and equivalence notions is adapted from concepts used by Fletcher et al. [10,11].

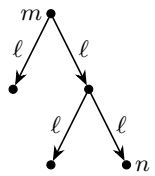

Fig. 2. A labeled tree that matches $\pi_1[\ell] \circ \ell \circ \pi_1[\ell] \circ \ell$ exactly.

$$
\begin{aligned}
[\![\pi_1[e]]\!]_{\mathcal{T}} &= \{(m,m) \mid \exists n\ (m,n) \in [\![e]\!]_{\mathcal{T}}\};\\
[\![\pi_2[e]]\!]_{\mathcal{T}} &= \{(n,n) \mid \exists m\ (m,n) \in [\![e]\!]_{\mathcal{T}}\};\\
[\![\overline{\pi}_j[e]]\!]_{\mathcal{T}} &= [\![\mathrm{id}]\!]_{\mathcal{T}} - [\![\pi_j[e]]\!]_{\mathcal{T}};\\
[\![e_1 \circ e_2]\!]_{\mathcal{T}} &= \{(m,n) \mid \exists z\ ((m,z) \in [\![e_1]\!]_{\mathcal{T}} \wedge (z,n) \in [\![e_2]\!]_{\mathcal{T}})\};\\
[\![e_1 \oplus e_2]\!]_{\mathcal{T}} &= [\![e_1]\!]_{\mathcal{T}} \oplus [\![e_2]\!]_{\mathcal{T}} \quad (\text{with } \oplus \in \{\cup, \cap, -\}).
\end{aligned}
$$

Notice that it suffices to consider converse $(^{-1})$ at the level of labels only. If an expression always evaluates to a subset of id, as is the case for projections and coprojections, then it is called a *node expression*.

Example 1. Consider the labeled tree in Figure 2. 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 $(\ell^{-1} \circ \ell) \cap \mathrm{di}$ and $(\ell^{-1} \circ \ell) - \mathrm{id}$ both return pairs of siblings in the tree.

For $k > 0$, we write $\mathcal{E}^k$ to represent k-fold composition of $\mathcal{E}$ and $\mathcal{E}^{-k}$ for its converse, we use $\mathcal{E}^0$ to denote id, $[\mathcal{E}]^+$ 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 $[\mathcal{E}^{-1}]^+ = \bigcup_{k>0} \mathcal{E}^{-k}$. Given $\mathcal{F} \subseteq \{\mathrm{di}, ^{-1}, \pi, \overline{\pi}, \cap, -\}$, $\mathcal{N}(\mathcal{F})$ denotes the relation algebra fragment in which only the atoms $\emptyset$, $\ell \in \Sigma$, and id, the operators $\circ$ and $\cup$, and all operators in $\mathcal{F}$ are allowed. In the above, we used π as shorthand for π_1 and π_2, and $\overline{\pi}$ as shorthand for $\overline{\pi}_1$ and $\overline{\pi}_2$.

Let q_1 and q_2 be expressions. We say that q_1 and q_2 are *path-equivalent*, denoted by $q_1 \equiv_{\mathrm{path}} q_2$, if, for every tree $\mathcal{T}$, $[\![q_1]\!]_{\mathcal{T}} = [\![q_2]\!]_{\mathcal{T}}$ and are *Boolean-equivalent*, denoted by $q_1 \equiv_{\mathrm{bool}} q_2$, if, for every tree $\mathcal{T}$, $[\![q_1]\!]_{\mathcal{T}} \neq \emptyset \iff [\![q_2]\!]_{\mathcal{T}} \neq \emptyset$. Let $z \in \{\mathrm{path}, \mathrm{bool}\}$. We say that the class of expressions $\mathcal{L}_1$ is *z-subsumed* by the class of expressions $\mathcal{L}_2$, denoted by $\mathcal{L}_1 \preceq_z \mathcal{L}_2$, if every expression in $\mathcal{L}_1$ is z-equivalent to an expression in $\mathcal{L}_2$. In this connection, the following rewrite rules can be used to express operators using other operators:

$$
\begin{aligned}
\pi_1[e] &= \pi_2[e^{-1}] = \overline{\pi}_j[\overline{\pi}_1[e]] = (e \circ e^{-1}) \cap \mathrm{id} = (e \circ (\mathrm{di} \cup \mathrm{id})) \cap \mathrm{id}\ (j \in \{1,2\});\\
\pi_2[e] &= \pi_1[e^{-1}] = \overline{\pi}_j[\overline{\pi}_2[e]] = (e^{-1} \circ e) \cap \mathrm{id} = ((\mathrm{di} \cup \mathrm{id}) \circ e) \cap \mathrm{id}\ (j \in \{1,2\});\\
\overline{\pi}_1[e] &= \overline{\pi}_2[e^{-1}] = \mathrm{id} - \pi_1[e];\\
\overline{\pi}_2[e] &= \overline{\pi}_1[e^{-1}] = \mathrm{id} - \pi_2[e];\\
e_1 \cap e_2 &= e_1 - (e_1 - e_2).
\end{aligned}
$$

Fig. 3. Trees $\mathcal{T}_1$, $\mathcal{T}_2$, and tree $\mathcal{T}_3$ from Example 5 and the proof of Proposition 7.

For $\mathcal{F} \subseteq \{\mathrm{di}, ^{-1}, \pi, \overline{\pi}, \cap, -\}$, $\overline{\mathcal{F}}$ denotes the closure of $\mathcal{F}$ under the rules above.[5]

Example 2. The equivalence $e_1 \cap e_2 \equiv_{\mathrm{path}} e_1 - (e_1 - e_2)$ is well-known. Hence, also $e_1 \cap e_2 \equiv_{\mathrm{bool}} e_1 - (e_1 - e_2)$. We also have $\pi_1[\ell] \equiv_{\mathrm{bool}} \ell \equiv_{\mathrm{bool}} \pi_2[\ell]$, but $\pi_1[\ell] \not\equiv_{\mathrm{path}} \ell$ and $\ell \not\equiv_{\mathrm{path}} \pi_2[\ell]$. Finally, let e be the expression as in Example 1. We have $e \equiv_{\mathrm{path}} \ell_1 \circ \ell_1^{-1} \circ \ell_2 \circ \ell_3 \circ \ell_3^{-1} \circ \ell_4$.

3 Subtree reductions

Most relation algebra fragments are able to detect obvious labeled branching in trees.

Example 3. Consider the expressions $e_1 = (\ell_1)^{-1} \circ \ell_2$ and $e_2 = \pi_1[\ell_1] \circ \pi_1[\ell_2]$, which are Boolean-equivalent. Clearly, for any tree $\mathcal{T}$ we have $[\![e_i]\!]_{\mathcal{T}} \neq \emptyset$, $i \in \{1, 2\}$ 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. Next, we look at which language fragments are able to detect branching if all branches are structurally identical. As a first step towards this goal, we derive limitations on the expressive power of relation algebra fragments, taking advantage of the simple structure of trees. Thereto, we introduce *subtree-reduction steps*.

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 $(m, n_1), \ldots, (m, n_{k+1}) \in \mathbf{E}(\ell)$ and the subtrees rooted at $n_1, \ldots, n_{k+1}$ are isomorphic, and then picking a node n_i, $1 \leq i \leq k+1$, and removing the subtree rooted at n_i.

Definition 4. *We say that a tree is* k-subtree-reducible *if we can apply a k-subtree-reduction step.*[6]

Example 5. Consider the unlabeled trees $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ in Figure 3. 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.

[5] The basic atoms and operators, $\emptyset$, $\ell \in \Sigma$, id, $\circ$, and $\cup$ are left implicit because they are assumed to be present in every fragment.

[6] The 1-subtree reductions bear a close relationship to the F+B-index and the F&B-index used for indexing the structure of tree data [20].

We now exhibit conditions under which the result of a relation algebra expression is invariant under subtree reduction at the Boolean level.

Proposition 6. *Let $\mathcal{F} \subseteq \{\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap, -\}$, e an expression in $\mathcal{N}(\mathcal{F})$, $\mathcal{T}$ a tree, and $\mathcal{T}'$ obtained from $\mathcal{T}$ by a k-subtree-reduction step. Each of the following conditions separately implies $[\![e]\!]_{\mathcal{T}} \neq \emptyset \iff [\![e]\!]_{\mathcal{T}'} \neq \emptyset$:*

(i) $k \geq 3$*;*
(ii) $k = 2$ *and* $\cap \notin \overline{\mathcal{F}}$*; and*
(iii) $k = 1$ *and* $\{\mathrm{di}, -\} \cap \mathcal{F} = \emptyset$*.*

Proof (sketch). Using k-pebble games [13,14,21], we can see that $[\![q]\!]_{\mathcal{T}} \neq \emptyset \iff [\![q]\!]_{\mathcal{T}'} \neq \emptyset$ if q is a query in FO[k], which is first-order logic restricted to k variables. Since the relation algebra and FO[3] path-subsume each other [12,24], (i) follows. In Statement (ii), $\mathcal{F} \subseteq \{\mathrm{di}, {}^{-1}, \pi, \overline{\pi}\}$. Hence, by a result of Hellings et al. [18, Theorem 6.1], (ii) also follows.[7] To prove (iii), let $\mathcal{T} = (\mathcal{V}, \Sigma, \mathbf{E})$ and $\mathcal{T}' = (\mathcal{V}', \Sigma', \mathbf{E}')$. Let $n_1, n_2 \in \mathcal{V}$ be the siblings in $\mathcal{T}$ such that $\mathcal{T}'$ is obtained from $\mathcal{T}$ by eliminating the subtree rooted at n_2. Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be the 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 a 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, for $m, n \in \mathcal{V}$, $(m, n) \in [\![e]\!]_{\mathcal{T}} \iff (f(m), f(n)) \in [\![e]\!]_{\mathcal{T}}$. By induction on the length of e, one can prove that, if $(m, n) \in \mathcal{V}_1 \times \mathcal{V}_2$ or $(m, n) \in \mathcal{V}_2 \times \mathcal{V}_1$, then $(m, n) \in [\![e]\!]_{\mathcal{T}} \implies (f(m), n) \in [\![e]\!]_{\mathcal{T}}$. Since $f = f^{-1}$, it then also follows that, if $(m, n) \in \mathcal{V}_1 \times \mathcal{V}_2$ or $(m, n) \in \mathcal{V}_2 \times \mathcal{V}_1$, then $(m, n) \in [\![e]\!]_{\mathcal{T}} \implies (m, f(n)) \in [\![e]\!]_{\mathcal{T}}$. A final induction on the length of e then yields that, for $m', n' \in \mathcal{V}'$, $(m', n') \in [\![e]\!]_{\mathcal{T}} \iff (m', n') \in [\![e]\!]_{\mathcal{T}'}$. Hence, $[\![e]\!]_{\mathcal{T}} \neq \emptyset \iff [\![e]\!]_{\mathcal{T}'} \neq \emptyset$, □

From the limitations imposed by Proposition 6 on the Boolean expressive power of the fragments considered, we deduce the following separation results:

Proposition 7. *Already on unlabeled trees, we have $\mathcal{N}(\mathrm{di}) \not\equiv_{\mathrm{bool}} \mathcal{N}({}^{-1}, \pi, \overline{\pi}, \cap)$, $\mathcal{N}({}^{-1}, -) \not\equiv_{\mathrm{bool}} \mathcal{N}({}^{-1}, \pi, \overline{\pi}, \cap)$, and $\mathcal{N}(\mathrm{di}, \cap) \not\equiv_{\mathrm{bool}} \mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi})$.*

Proof. Consider the unlabeled trees $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ in Example 5. Since $\mathcal{T}_1$ can be obtained by a 1-subtree-reduction on $\mathcal{T}_2$, we have, by Proposition 6 (iii), that, for every e in $\mathcal{N}({}^{-1}, \pi, \overline{\pi}, \cap)$, $[\![e]\!]_{\mathcal{T}_2} \neq \emptyset \iff [\![e]\!]_{\mathcal{T}_1} \neq \emptyset$. Now consider $e_1 = \mathcal{E} \circ \mathrm{di} \circ \mathrm{di} \circ \mathcal{E}$ in $\mathcal{N}(\mathrm{di})$ and $e_2 = (\mathcal{E}^{-1} \circ \mathcal{E}) - \mathrm{id}$ in $\mathcal{N}({}^{-1}, -)$. We have $[\![e_1]\!]_{\mathcal{T}_2} \neq \emptyset$ and $[\![e_2]\!]_{\mathcal{T}_2} \neq \emptyset$, while $[\![e_1]\!]_{\mathcal{T}_1} = [\![e_2]\!]_{\mathcal{T}_1} = \emptyset$, establishing the first and second separations. Since $\mathcal{T}_2$ can be obtained by a 2-subtree-reduction on $\mathcal{T}_3$, we have, by Proposition 6 (ii), that, for every e in $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi, \overline{\pi})$, $[\![e]\!]_{\mathcal{T}_3} \neq \emptyset \iff [\![e]\!]_{\mathcal{T}_2} \neq \emptyset$. Now consider $e_3 = (((\mathrm{di} \circ \mathcal{E}) \cap \mathrm{di}) \circ ((\mathrm{di} \circ \mathcal{E}) \cap \mathrm{di})) \cap \mathrm{di}$ in $\mathcal{N}(\mathrm{di}, \cap)$. We have $[\![e_3]\!]_{\mathcal{T}_3} \neq \emptyset$, while $[\![e_3]\!]_{\mathcal{T}_2} = \emptyset$, establishing the third separation. □

The proof of Proposition 7 relies on languages being able to distinguish at least one, two, or three structurally equivalent children of a node. To do so, the proof uses minimal languages that satisfy the conditions of Proposition 6. Hence, the classification provided by k-subtree-reductions is strict.

[7] Notice that in the formalism of Hellings et al. [18], projection is considered to be a standard operator.

4 The power of diversity

Relation algebra expressions without diversity can only inspect a local neighborhood around a given node. With respect to path queries, this puts obvious limitations on the expressive power of language fragments that do not contain diversity. With respect to Boolean queries, the situation is more subtle. To study this in more detail, we first define the notion of locality:

Definition 8. *Given a tree, and disregarding the direction of its edges, the* distance *between two nodes is the number of edges on the unique shortest path between them. 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, $(m,n) \in [\![q]\!]_{\mathcal{T}} \iff (m,n) \in [\![q]\!]_{\mathcal{T}'}$, 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.*

By an induction on their length, it can be shown that all expressions in $\mathcal{N}(^{-1}, \pi, \overline{\pi}, \cap, -)$ are local.

4.1 Adding diversity to local fragments

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.

Proposition 9. *Already on unlabeled trees, $\mathcal{N}(\mathrm{di}, \cap) \not\preceq_{\mathrm{bool}} \mathcal{N}(^{-1}, \pi, \overline{\pi}, \cap, -)$.*

Proof. By the rewrite rules at the end of Section 2, $\mathcal{N}(\mathrm{di}, \pi, \cap) \preceq_{\mathrm{path}} \mathcal{N}(\mathrm{di}, \cap)$. Consider the expression $e = P_{2,\neg r} \circ \mathrm{di} \circ P_{2,\neg r}$ in which $P_{2,\neg r} = \pi_2[\mathcal{E}] \circ P_2$, $P_2 = \pi_1[S_2]$, and $S_2 = (\mathcal{E} \circ \mathrm{di}) \cap \mathcal{E}$. The expression e selects node pairs among distinct non-root nodes such that each node in the pair has at least two distinct children. Now, assume there exists an expression e' in $\mathcal{N}(^{-1}, \pi, \overline{\pi}, \cap, -)$ such that $e \equiv_{\mathrm{bool}} e'$. Since e' is local, we know there exists $k \geq 0$ such that e' satisfies Definition 8. Now consider the trees $\mathcal{T}$ and $\mathcal{T}'$ shown in Figure 4. Clearly, $[\![e]\!]_{\mathcal{T}} \supseteq \{(m_1, m_2)\} \neq \emptyset$ and $[\![e]\!]_{\mathcal{T}'} = \emptyset$. By $e \equiv_{\mathrm{bool}} e'$, 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 $e \equiv_{\mathrm{bool}} e'$. Hence, no expression in $\mathcal{N}(^{-1}, \pi, \overline{\pi}, \cap, -)$ is Boolean-equivalent to e. □

We can use a similar locality argument for two more separations:

Proposition 10. *Already on chains, we have $\mathcal{N}(\mathrm{di}, ^{-1}) \not\preceq_{\mathrm{bool}} \mathcal{N}(^{-1}, \pi, \overline{\pi}, \cap, -)$ and $\mathcal{N}(\mathrm{di}, \pi) \not\preceq_{\mathrm{bool}} \mathcal{N}(^{-1}, \pi, \overline{\pi}, \cap, -)$.*

Proof. Consider the path-equivalent expressions $e_1 = (\ell \circ \ell^{-1}) \circ \mathrm{di} \circ (\ell \circ \ell^{-1})$ in $\mathcal{N}(\mathrm{di}, ^{-1})$ and $e_2 = \pi_1[\ell] \circ \mathrm{di} \circ \pi_1[\ell]$ in $\mathcal{N}(\mathrm{di}, \pi)$, and let e be either e_1 or e_2. Now, assume there exists an expression e' in $\mathcal{N}(^{-1}, \pi, \overline{\pi}, \cap, -)$ such that $e \equiv_{\mathrm{bool}} e'$.

Fig. 4. Trees $\mathcal{T}$ and $\mathcal{T}'$ in the proof of Proposition 9. The symbol $\rightsquigarrow$ represents a chain of k edges, with k as in that proof.

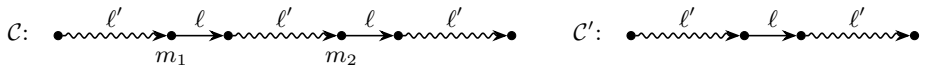

Fig. 5. Chains $\mathcal{C}$ and $\mathcal{C}'$ in the proof of Proposition 10. The symbol $\rightsquigarrow$ represents a chain of $2k$ edges all labeled ℓ', with k as in that proof.

Since e' is local, we know there exists $k \geq 0$ such that e' satisfies Definition 8. Now consider the chains $\mathcal{C}$ and $\mathcal{C}'$ shown in Figure 5. Clearly, $[\![e]\!]_{\mathcal{C}} \supseteq \{(m_1, m_2)\} \neq \emptyset$. Hence, $[\![e']\!]_{\mathcal{C}} \neq \emptyset$. By $e \equiv_{\text{bool}} e'$, we must have $[\![e']\!]_{\mathcal{T}} \neq \emptyset$. Let $(m, n) \in [\![e']\!]_{\mathcal{T}}$. Notice that every subchain of $\mathcal{C}$ containing all nodes at distance at most k from some given node is a subchain of $\mathcal{C}$ of length at most $2k$. Since each such subchain of $\mathcal{C}$ is isomorphic to some subchain of $\mathcal{C}'$, we may conclude that $[\![e']\!]_{\mathcal{C}'} \neq \emptyset$. However, $[\![e]\!]_{\mathcal{C}'} = \emptyset$, contradicting $e \equiv_{\text{bool}} e'$. Hence, no expression in $\mathcal{N}(^{-1}, \pi, \overline{\pi}, \cap, -)$ is Boolean-equivalent to e. □

Observe that the separation $\mathcal{N}(\text{di}, \cap) \not\equiv_{\text{bool}} \mathcal{N}(^{-1}, \pi, \overline{\pi}, \cap, -)$ also holds on chains, because the expression $e_3 = (\ell \circ \text{di} \cap \text{id}) \circ \text{di} \circ (\ell \circ \text{di} \cap \text{id})$ is path-equivalent to e_1 and e_2 in the proof of Proposition 10.

4.2 Adding other operators to non-local fragments

The erratic behavior of diversity in the non-local relation algebra fragments on trees (allowing one to jump from any node to any other node in a tree) makes studying the expressive power of these fragments inherently difficult. Luckily, we can obtain several separation results by studying these fragments on chains.

For local expressions on chains, we have the following:

Lemma 11. *Let $\mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}, \cap, -\}$, and e be a union-free expression in $\mathcal{N}(\mathcal{F})$.*[8] *There exists $k \geq 0$ such that, for every chain $\mathcal{C}$, $[\![e]\!]_{\mathcal{C}} \subseteq [\![\mathcal{E}^k]\!]_{\mathcal{C}}$ or $[\![e]\!]_{\mathcal{C}} \subseteq [\![\mathcal{E}^{-k}]\!]_{\mathcal{C}}$.*

Lemma 11 simplifies the reasoning about local expressions on chains. In addition, diversity on chains can be expressed using the ancestor axis and the descendant axis, as $\text{di} = [\mathcal{E}]^+ \cup [\mathcal{E}^{-1}]^+$. While the descendant and ancestor axes are not operators of the fragments considered in this paper, we can still use them in intermediate steps to rewrite expressions that contain di. This in turn allows us to simplify projection terms that contain di, as we show next.

[8] Observe that, in relation algebra expressions, unions can always be pushed out to the outermost level.

Lemma 12. *Let $\mathcal{F} \subseteq \{\mathrm{di}, {}^{-1}, \pi\}$ and $\pi_j[e]$, $j \in \{1,2\}$, 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 $\mathrm{di} \equiv_{\mathrm{path}} [\mathcal{E}]^+ \cup [\mathcal{E}^{-1}]^+$. 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 built over the atoms id, $\mathcal{E}$, $\mathcal{E}^{-1}$, $[\mathcal{E}]^+$, and $[\mathcal{E}^{-1}]^+$, using the operators $\circ$ and π_1.[9] We shall call such expressions e' *normal* in the remainder of this proof. So, it remains to show that Lemma 12 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 12 holds for expressions $\pi_1[e'']$, with e'' a normal expression containing at most i operators, $i \geq 0$, and let $e = \pi_1[e']$ 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, $e \equiv_{\mathrm{path}} \pi_1[e'_1]$, and Lemma 12 holds for e by the induction hypothesis. In the second case, we have that $e = \pi_1[e'_1 \circ e'_2] \equiv_{\mathrm{path}} \pi_1[e'_1 \circ \pi_1[e'_2]]$. 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 again 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

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

to conclude that Lemma 12 holds for e.

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

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

to conclude that Lemma 12 holds for e in this case, too. □

[9] Recall from Section 2 that we need to consider converse only at the level of edges.

Example 13. Consider the expression $e = \pi_1[\mathrm{di} \circ \mathcal{E} \circ \mathcal{E}]$. We have

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

As Example 13 shows, we can use Lemma 12 to partially eliminate diversity from non-local expressions on unlabeled chains, and then use locality-based arguments on subexpressions to establish the following separations:

Proposition 14. *Already on unlabeled chains,* $\mathcal{N}(\mathrm{di}, \cap) \not\equiv_{\mathrm{path}} \mathcal{N}(\mathrm{di}, {}^{-1}, \pi)$, $\mathcal{N}({}^{-1}) \not\equiv_{\mathrm{path}} \mathcal{N}(\mathrm{di}, \pi)$, *and* $\mathcal{N}(\pi) \not\equiv_{\mathrm{path}} \mathcal{N}(\mathrm{di})$.

Proof. Consider the expression $e = (\mathrm{di} \circ \mathcal{E}) \cap \mathrm{di}$ in $\mathcal{N}(\mathrm{di}, \cap)$. 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 $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi)$ such that, on unlabeled chains, $e \equiv_{\mathrm{path}} e'$. Since e is non-local, e' must be non-local, too, hence, it must contain diversity. Using Lemma 12, we can rewrite e' into a union of terms each of which is a composition of units of the form id, di, $\mathcal{E}$, $\mathcal{E}^{-1}$, $\pi_1[\mathcal{E}^v]$, or $\pi_2[\mathcal{E}^w]$, $v, w \geq 0$. Let $t = t_1 \circ \cdots \circ t_n$ be such a term in which at least one unit is diversity (di). Since on chains $\mathrm{di} \equiv_{\mathrm{path}} [\mathcal{E}]^+ \cup [\mathcal{E}^{-1}]^+$, t is path-equivalent to the infinite union $\bigcup_{k_1,\ldots,k_n \neq 0} t_{1k_1} \circ \cdots \circ t_{nk_n}$ in which $t_{ik_i} = t_i$ if $t_i \neq \mathrm{di}$ and $t_{ik_i} = \mathcal{E}^{k_i}$ if $t_i = \mathrm{di}$, $1 \leq i \leq n$. For a term $t_{1k_1} \circ \cdots \circ t_{nk_n}$ in this infinite union, we define $\exp(t_{1k_1} \circ \cdots \circ t_{nk_n}) = \sum_{1 \leq i \leq n} \exp(t_{ik_i})$, where $\exp(t_{ik_i}) = 1$ if $t_{ik_i} = \mathcal{E}$; $\exp(t_{ik_i}) = -1$ if $t_{ik_i} = \mathcal{E}^{-1}$; $\exp(t_{ik_i}) = k_i$ if $t_{ik_i} = \mathcal{E}^{k_i}$; and $\exp(t_{ik_i}) = 0$ otherwise. Since t contains at least one diversity unit, the set $\{\exp(t_{1k_1} \circ \cdots \circ t_{nk_n}) \mid k_1, \ldots, k_n \neq 0\}$ covers all integer numbers with at most one exception (there is exactly one exception if t contains exactly one diversity unit). We can therefore choose a term $t' = t_{1k_1} \circ \cdots \circ t_{nk_n}$ for which $\exp(t') = 0$ or $\exp(t') = 1$. Now, choose an unlabeled chain $\mathcal{C}$ which is sufficiently long to ensure that for the local expression t' in $\mathcal{N}({}^{-1}, \pi)$, $[\![t']\!]_{\mathcal{C}} \neq \emptyset$. Then, $[\![t']\!]_{\mathcal{C}}$ contains either an identical node pair (if $\exp(t') = 0$) or an edge (if $\exp(t') = 1$). Hence, by construction, $[\![e']\!]_{\mathcal{C}}$ contains either an identical node pair or an edge, contradicting $e \equiv_{\mathrm{path}} e'$. Hence, no expression in $\mathcal{N}(\mathrm{di}, {}^{-1}, \pi)$ is path-equivalent to e.

Using similar arguments, we can prove that $\mathcal{E}^{-1}$ cannot be expressed in $\mathcal{N}(\mathrm{di}, \pi)$ and that $\pi_1[\mathcal{E}]$ and $\pi_2[\mathcal{E}]$ cannot be expressed in $\mathcal{N}(\mathrm{di})$ to establish the other two statements of Proposition 14. □

5 Brute-force results

Using a brute-force approach in the style of Fletcher et al. [10], we establish several separations, both at the path and Boolean levels. At the core of these brute-force results is the observation than one can effectively compute the set of query results obtainable by queries in some relation algebra fragment $\mathcal{N}(\mathcal{F})$, $\mathcal{F} \subseteq \{\mathrm{di}, {}^{-1}, \pi, \overline{\pi}, \cap, -\}$, on a given graph. We refer to Fletcher et al. [10] and Hellings [15] for further details.

Fig. 6. The unlabeled tree $\mathcal{T}$ in the proof of Proposition 15.

For path separations between languages $\mathcal{L}_1$ and $\mathcal{L}_2$, we may conclude that $\mathcal{L}_1 \not\equiv_{\text{path}} \mathcal{L}_2$ if there exists a query q in $\mathcal{L}_1$ and a tree $\mathcal{T}$ such that no query in $\mathcal{L}_2$ evaluates to $[\![q]\!]_{\mathcal{T}}$. Using this approach, we prove the following.

Proposition 15. *Already on unlabeled trees, we have* $\mathcal{N}(^{-1}) \not\equiv_{\text{path}} \mathcal{N}(\text{di}, \pi, \overline{\pi})$ *and* $\mathcal{N}(\pi) \not\equiv_{\text{path}} \mathcal{N}(^{-1})$.

Proof. Consider the expressions $e_1 = \mathcal{E}^{-1}$ and $e_2 = \pi_1[\mathcal{E}] \circ \pi_2[\mathcal{E}]$, and let $\mathcal{T}$ be the tree in Figure 6. An exhaustive search reveals that no expression in $\mathcal{N}(\text{di}, \pi, \overline{\pi})$ evaluates to $[\![e_1]\!]_{\mathcal{T}}$ and no expression in $\mathcal{N}(^{-1})$ evaluates to $[\![e_2]\!]_{\mathcal{T}}$. □

At the Boolean level, the key notion in the brute-force approach is the ability to *distinguish* a pair of trees. We say that a query q distinguishes a pair of trees $\mathcal{T}_1$ and $\mathcal{T}_2$ if $[\![q]\!]_{\mathcal{T}_1} = \emptyset$ and $[\![q]\!]_{\mathcal{T}_2} \neq \emptyset$, or vice versa. Given two languages $\mathcal{L}_1$ and $\mathcal{L}_2$, we may conclude that $\mathcal{L}_1 \not\equiv_{\text{bool}} \mathcal{L}_2$ if we can find a query q in $\mathcal{L}_1$ and a pair of trees $\mathcal{T}_1$ and $\mathcal{T}_2$, indistinguishable by any query in $\mathcal{L}_2$, but distinguishable by q. Using this approach, we prove the following.

Proposition 16. *Already on unlabeled trees, we have* $\mathcal{N}(\text{di}, ^{-1}) \not\equiv_{\text{bool}} \mathcal{N}(\text{di})$, $\mathcal{N}(\text{di}, \pi) \not\equiv_{\text{bool}} \mathcal{N}(\text{di})$, $\mathcal{N}(\text{di}, ^{-1}, \cap) \not\equiv_{\text{bool}} \mathcal{N}(\text{di}, \pi, \overline{\pi}, \cap, -)$, *and* $\mathcal{N}(\text{di}, \pi) \not\equiv_{\text{bool}} \mathcal{N}(\text{di}, ^{-1})$.

Proof. Consider the following four expressions:

$$e_1 = (\mathcal{E}^{-2} \circ \mathcal{E}) \circ \text{di} \circ (\mathcal{E}^{-1} \circ \mathcal{E}^2);$$
$$e_2 = (\mathcal{E} \circ \pi_1[\mathcal{E}]) \circ \text{di} \circ (\pi_2[\mathcal{E}] \circ \mathcal{E});$$
$$e_3 = (((\mathcal{E}^{-1} \circ \mathcal{E}) \cap \text{di}) \circ ((\mathcal{E}^{-1} \circ \mathcal{E}) \cap \text{di})) \cap \text{di};$$
$$e_4 = (\mathcal{E}^2 \circ \mathcal{E}^{-1}) \circ \text{di} \circ \pi_1[\mathcal{E}^{-1} \circ \mathcal{E}^2] \circ \text{di} \circ \pi_1[\mathcal{E}^{-1} \circ \mathcal{E}^2] \circ \text{di} \circ (\mathcal{E} \circ \mathcal{E}^{-2}),$$

and let $(\mathcal{T}_i, \mathcal{T}_i')$, $1 \leq i \leq 4$, be the trees in Figure 7. We have $[\![e_i]\!]_{\mathcal{T}_i} = \emptyset$ and $[\![e_i]\!]_{\mathcal{T}_i'} \neq \emptyset$, and $[\![e_2]\!]_{\mathcal{T}_1} = \emptyset$ and $[\![e_1]\!]_{\mathcal{T}_1'} \neq \emptyset$. Observe that e_4 is in $\mathcal{N}(\text{di}, ^{-1}, \pi)$, but, by Proposition 34, e_4 is Boolean-equivalent to an expression in $\mathcal{N}(\text{di}, \pi)$. An exhaustive search reveals that no expression in $\mathcal{N}(\text{di})$ can distinguish $\mathcal{T}_1 = \mathcal{T}_2$ from $\mathcal{T}_1' = \mathcal{T}_2'$; no expression in $\mathcal{N}(\text{di}, \pi, \overline{\pi}, \cap, -)$ can distinguish $\mathcal{T}_3$ from $\mathcal{T}_3'$; and no expression in $\mathcal{N}(\text{di}, ^{-1})$ can distinguish $\mathcal{T}_4$ from $\mathcal{T}_4'$. □

6 Collapse results

In Sections 3, 4, and 5, we focused on separation results. Here, we focus on collapse results. The key tool to prove these are what we call *condition tree queries*, a generalization of the tree queries of Wu et al. [26], which were used to prove Proposition 30 (see Section 7 on related work).

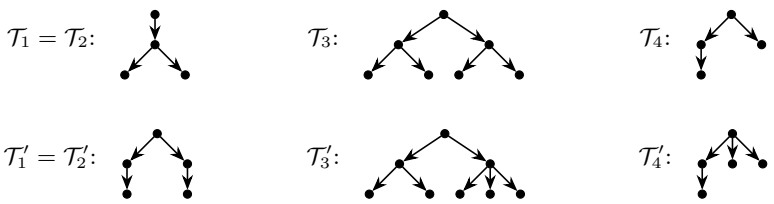

Fig. 7. Pairs of unlabeled trees $(\mathcal{T}_i, \mathcal{T}_i')$, $1 \leq i \leq 4$, in the proof of Proposition 16.

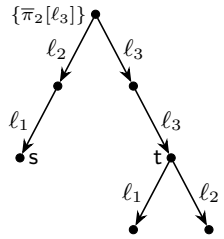

Fig. 8. The condition tree query of Example 18.

6.1 Condition tree queries

We first define *condition tree queries* syntactically and semantically:

Definition 17. *A* condition tree query *is a tuple* $\mathcal{Q} = (\mathcal{T}, C, \mathsf{s}, \mathsf{t}, \gamma)$*, where* $\mathcal{T} = (\mathcal{V}, \Sigma, \mathbf{E})$ *is a 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* $\{c \mid (n, c) \in \gamma\}$*.*

Let $\mathcal{T}' = (\mathcal{V}', \Sigma, \mathbf{E}')$ *be a tree. Then,* $[\![\mathcal{Q}]\!]_{\mathcal{T}'}$ *consists of all the 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:*

(i) $f(\mathsf{s}) = m$ *and* $f(\mathsf{t}) = n$*;*
(ii) for all $v \in \mathcal{V}$ *and* $c \in \gamma(v)$*,* $(f(v), f(v)) \in [\![c]\!]_{\mathcal{T}'}$*; and*
(iii) for all $\ell \in \Sigma$ *and* $(v, w) \in \mathbf{E}(\ell)$*,* $(f(v), f(w)) \in \mathbf{E}'(\ell)$*.*

We slightly extend Definition 17 to allow the *empty condition tree* where $\mathcal{V} = \emptyset$ and s and t have some null value. On every tree, the empty condition tree evaluates to the empty set.

Example 18. The condition tree query $\mathcal{Q}$ in Figure 8 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 condition $\overline{\pi}_2[\ell_3]$; (3) from there, go down via two edges labeled ℓ_3, after which we arrive at t; and (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 $\ell_1^{-1} \circ \ell_2^{-1} \circ \overline{\pi}_2[\ell_3] \circ \ell_3 \circ \ell_3 \circ \pi_1[\ell_1] \circ \pi_1[\ell_2]$, in $\mathcal{N}(^{-1}, \overline{\pi})$.

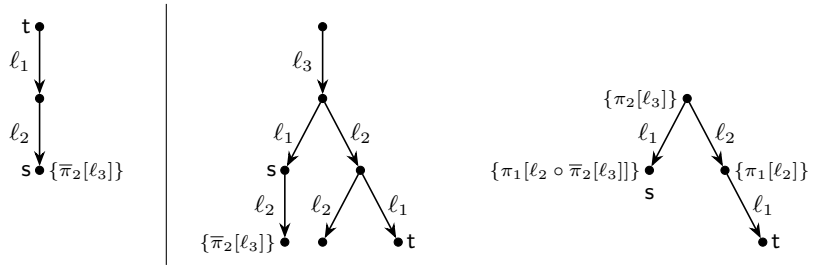

Fig. 9. The condition tree query on the *left* is up-down. The condition tree query in the *middle* is not, but this query is path-equivalent to the up-down query on the right.

In the remainder of this subsection, we formalize the relationship between condition tree queries and relation algebra expressions exhibited in Example 18. Thereto, let $\mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}\}$, and let $\mathbf{Q}_{\text{tree}}(\mathcal{F})$ be the class of all condition tree queries in which node conditions are restricted to union-free expressions in $\mathcal{N}(\mathcal{F})$. We claim that, for $\overline{\mathcal{F}} = \{^{-1}, \pi\}$ or $\overline{\mathcal{F}} = \{^{-1}, \pi, \overline{\pi}\}$, $\mathbf{Q}_{\text{tree}}(\mathcal{F})$ and the class of all union-free expressions in $\mathcal{N}(\mathcal{F})$ path-subsume each other.

Using a straightforward rewriting argument, we can show the following:

Proposition 19. *Let $\mathcal{F} \subseteq \{^{-1}, \pi, \overline{\pi}\}$, and e be a union-free expression in $\mathcal{N}(\mathcal{F})$. There exists a condition tree query $\mathcal{Q}$ in $\mathbf{Q}_{\text{tree}}(\mathcal{F})$ such that $e \equiv_{\text{path}} \mathcal{Q}$.*

For the translation in the other direction, we introduce *up-down queries*:

Definition 20. *An* up-down query *is a condition tree query $\mathcal{Q} = (\mathcal{T}, C, \mathsf{s}, \mathsf{t}, \gamma)$ in which all edges of $\mathcal{T}$ are on the unique path from s to t not taking into account the direction of the edges.*

Example 21. 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 9, *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 the parts of the tree traversal described by the *middle* query that are not on the path from source to target 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 21, 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:

Lemma 22. *Let $\{\pi\} \subseteq \overline{\mathcal{F}} \subseteq \{^{-1}, \overline{\pi}, \pi\}$, and $\mathcal{Q}$ be a condition tree query in $\mathbf{Q}_{\text{tree}}(\mathcal{F})$. There exists an up-down query $\mathcal{Q}'$ in $\mathbf{Q}_{\text{tree}}(\mathcal{F})$ such that $\mathcal{Q} \equiv_{\text{path}} \mathcal{Q}'$.*

As also illustrated in Example 21, 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:

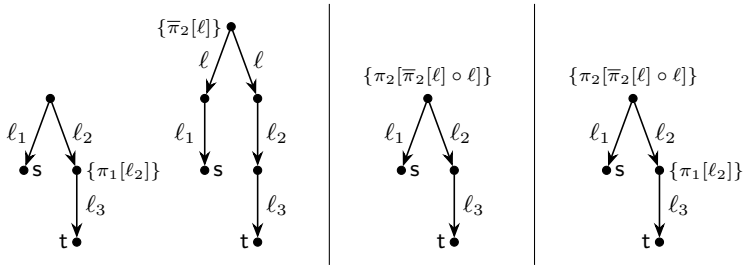

Fig. 10. The step-wise computation of the intersection of the two up-down queries on the *left* eventually results in the up-down query on the *right*.

Lemma 23. *Let* $\{^{-1}\} \subseteq \mathcal{F} \subseteq \{^{-1}, \overline{\pi}, \pi\}$*, and* $\mathcal{Q}$ *be an up-down query in* $\mathbf{Q}_{\text{tree}}(\mathcal{F})$*. There exists a union-free expression* e *in* $\mathcal{N}(\mathcal{F})$ *such that* $\mathcal{Q} \equiv_{\text{path}} e$*.*

Finally, combining Lemmas 22 and 23 yields the following:

Proposition 24. *Let* $\{^{-1}, \pi\} \subseteq \overline{\mathcal{F}} \subseteq \{^{-1}, \pi, \overline{\pi}\}$*, and* $\mathcal{Q}$ *be a condition tree query in* $\mathbf{Q}_{\text{tree}}(\mathcal{F})$*. There exists a union-free expression in* $\mathcal{N}(\mathcal{F})$ *such that* $\mathcal{Q} \equiv_{\text{path}} e$*.*

6.2 Adding intersection to local fragments

Hellings et al. [16,17] 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 31). Here, we show that this result actually holds for *all* local relation algebra fragments. As a first step, consider the following example:

Example 25. Suppose we want to compute the intersection of the two up-down queries in Figure 10, *left*. Since the two up-down queries have different heights, 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*.

We now generalize Example 25:

Proposition 26. *Let* $\{\pi\} \subseteq \overline{\mathcal{F}} \subseteq \{^{-1}, \pi, \overline{\pi}\}$*, and* $\mathcal{Q}_1$ *and* $\mathcal{Q}_2$ *be condition tree queries in* $\mathbf{Q}_{\text{tree}}(\mathcal{F})$*. There exists a condition tree query* $\mathcal{Q}$ *in* $\mathbf{Q}_{\text{tree}}(\mathcal{F})$ *such that, for every tree* $\mathcal{T}$*,* $[\![\mathcal{Q}]\!]_{\mathcal{T}} = [\![\mathcal{Q}_1]\!]_{\mathcal{T}} \cap [\![\mathcal{Q}_2]\!]_{\mathcal{T}}$*.*

Proof (sketch). By Lemma 22, we may assume that $\mathcal{Q}_1 = (\mathcal{T}_1, C_1, \mathtt{s}_1, \mathtt{t}_1, \gamma_1)$ and $\mathcal{Q}_2 = (\mathcal{T}_2, C_2, \mathtt{s}_2, \mathtt{t}_2, \gamma_2)$ are up-down queries as in Figure 11. Let $\mathcal{T}'$ be an arbitrary tree. If $u_2 - d_2 \neq u_1 - d_1$, then, obviously $[\![\mathcal{Q}_1]\!]_{\mathcal{T}'} \cap [\![\mathcal{Q}_2]\!]_{\mathcal{T}'} = \emptyset$. Thus, assume $u_2 - d_2 = u_1 - d_1$, or, equivalently, $u_2 - u_1 = d_2 - d_1$. We distinguish two cases:

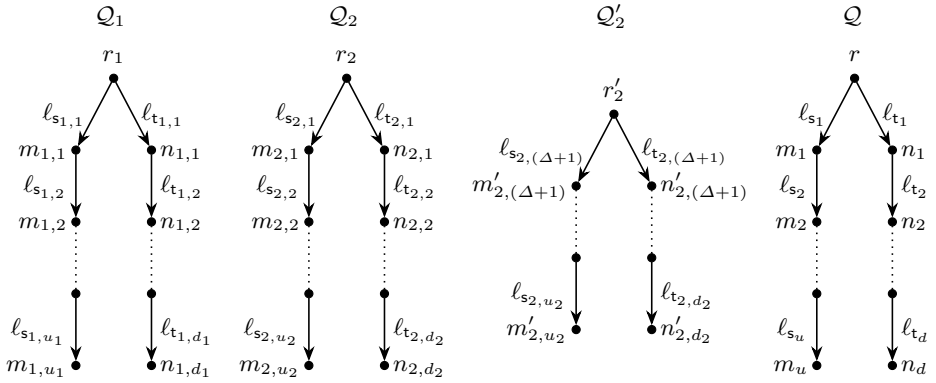

Fig. 11. Up-down queries $\mathcal{Q}_1$, $\mathcal{Q}_2$, $\mathcal{Q}'_2$ and $\mathcal{Q}$ in the proof of Proposition 26.

1. $u_1 \neq u_2$. By symmetry, assume $u_2 > u_1$. We write $\Delta = u_2 - u_1 = d_2 - d_1$. To find a pair of nodes of $\mathcal{T}'$ common to $[\![\mathcal{Q}_1]\!]_{\mathcal{T}'}$ and $[\![\mathcal{Q}_2]\!]_{\mathcal{T}'}$, it is imperative that for all i, $1 \leq i \leq \Delta$, $m_{2,i}$ and $n_{2,i}$ are mapped to the same node of $\mathcal{T}$. Hence, if for some i, $1 \leq i \leq \Delta$, $\ell_{\mathsf{s}_{2,i}} \neq \ell_{\mathsf{t}_{2,i}}$, $[\![\mathcal{Q}_1]\!]_{\mathcal{T}'} \cap [\![\mathcal{Q}_2]\!]_{\mathcal{T}'} = \emptyset$. Thus, assume for all i, $1 \leq i \leq \Delta$, that $\ell_{\mathsf{s}_{2,i}} = \ell_{\mathsf{t}_{2,i}}$. Then, $[\![\mathcal{Q}_1]\!]_{\mathcal{T}'} \cap [\![\mathcal{Q}_2]\!]_{\mathcal{T}'} = [\![\mathcal{Q}_1]\!]_{\mathcal{T}'} \cap [\![\mathcal{Q}'_2]\!]_{\mathcal{T}'}$, where $\mathcal{Q}'_2 = (\mathcal{T}'_2, C'_2, \mathsf{s}'_2, \mathsf{t}'_2, \gamma'_2)$ is as in Figure 11, with

$$\gamma'_2(r'_2) = \pi_2[\gamma_2(r_2) \circ \ell_{\mathsf{s}_{2,1}} \circ \gamma_2(m_{2,1}) \circ \gamma_2(n_{2,1}) \circ \cdots \circ \ell_{\mathsf{s}_{2,\Delta}} \circ \gamma_2(m_{2,\Delta}) \circ \gamma_2(n_{2,\Delta})],$$

in which $\gamma_2(v)$ is a shorthand for the composition of the node expressions in $\gamma(v)$.[10] For all other nodes v' of $\mathcal{T}'_2$, $\gamma'_2(v') = \gamma_2(v)$, v being the node of $\mathcal{T}_2$ corresponding to v'. Notice that the left (an hence also the right) branches of $\mathcal{Q}_1$ and $\mathcal{Q}'_2$ have equal length, which allows us to apply the next case on $\mathcal{Q}_1$ and $\mathcal{Q}'_2$.

2. $u_1 = u_2 = u$, and hence $d_1 = d_2 = d$. To find a pair of nodes of $\mathcal{T}'$ common to $[\![\mathcal{Q}_1]\!]_{\mathcal{T}'}$ and $[\![\mathcal{Q}_2]\!]_{\mathcal{T}'}$, it is imperative that corresponding nodes of $\mathcal{T}_1$ and $\mathcal{T}_2$ are mapped to the same node of $\mathcal{T}'$. Hence, if for some i, $1 \leq i \leq u$, $\ell_{\mathsf{s}_{1,i}} \neq \ell_{\mathsf{s}_{2,i}}$, or for some j, $1 \leq j \leq d$, $\ell_{\mathsf{t}_{1,i}} \neq \ell_{\mathsf{t}_{2,i}}$, $[\![\mathcal{Q}_1]\!]_{\mathcal{T}'} \cap [\![\mathcal{Q}_2]\!]_{\mathcal{T}'} = \emptyset$. Thus, assume for all i, $1 \leq i \leq u$, that $\ell_{\mathsf{s}_{1,i}} = \ell_{\mathsf{s}_{2,i}} = \ell_{\mathsf{s}_i}$, and, for all j, $1 \leq j \leq d$, that $\ell_{\mathsf{t}_{1,i}} = \ell_{\mathsf{t}_{2,i}} = \ell_{\mathsf{t}_i}$. Then, $[\![\mathcal{Q}_1]\!]_{\mathcal{T}'} \cap [\![\mathcal{Q}_2]\!]_{\mathcal{T}'} = [\![\mathcal{Q}]\!]_{\mathcal{T}'}$, where $\mathcal{Q} = (\mathcal{T}, C_1 \cup C_2, \mathsf{s}, \mathsf{t}, \gamma)$ is as in Figure 11, where, for all nodes v of $\mathcal{T}$, $\gamma(v) = \gamma_1(v_1) \circ \gamma_2(v_2)$, v_1 and v_2 being the nodes of $\mathcal{T}_1$ and $\mathcal{T}_2$ corresponding to v. □

Propositions 19, 24, and 26 now yield the following:

Proposition 27. *For* $\{^{-1}, \pi\} \subseteq \overline{\mathcal{F}} \subseteq \{^{-1}, \pi, \overline{\pi}, \cap\}$, $\mathcal{N}(\mathcal{F}) \preceq_{\text{path}} \mathcal{N}(\mathcal{F} - \{\cap\})$.

6.3 The Boolean equivalence of projection and converse

From a result by Fletcher et al. [10,11] (Proposition 34 in Section 7 on related work), it follows that $\mathcal{N}(^{-1}) \preceq_{\text{bool}} \mathcal{N}(\pi)$. Here, we also prove the other direction:

[10] Observe that the composition of node expressions is associative and that this composition is path-equivalent to the intersection of node expressions.

Proposition 28. $\mathcal{N}(\pi) \preceq_{\text{bool}} \mathcal{N}(^{-1})$.

Proof. Let e be an expression in $\mathcal{N}(\pi)$. By a result of Wu et al. [26, Theorem 4.1], there exists a condition-free condition tree query $\mathcal{Q} = (\mathcal{T}, C, \mathsf{s}, \mathsf{t}, \gamma)$ in $\mathbf{Q}_{\text{tree}}()$ such that $e \equiv_{\text{path}} \mathcal{Q}$.[11] Let r be the root of $\mathcal{T}$, and $\mathcal{Q}_r = (\mathcal{T}, C, r, r, \gamma)$. Obviously, $\mathcal{Q} \equiv_{\text{bool}} \mathcal{Q}_r$. Because the target node is now the root of $\mathcal{T}$, the translation from $\mathcal{Q}_r$ to a path-equivalent up-down query (Lemma 22) only requires the first projection. Hence, there exists an up-down query $\mathcal{Q}'_r = (\mathcal{T}', C', r', r', \gamma')$ in $\mathbf{Q}_{\text{tree}}(\pi_1)$ such that $\mathcal{Q}_r \equiv_{\text{path}} \mathcal{Q}'_r$. Since source and target coincide in $\mathcal{T}'$, r' is necessarily the only node of $\mathcal{T}'$. Hence, $\mathcal{Q}'_r$ is path equivalent to the composition of the node expressions in $\gamma(r')$, which is in $\mathcal{N}(\pi_1)$. Now, a projection expression in $\mathcal{N}(\pi_1)$ can always be rewritten in the form $\pi_1[e] = \pi_1[\ell_1 \circ \pi_1[e_1] \circ \cdots \ell_n \circ \pi_1[e_n]]$, with $e_1, \ldots, e_n$ in $\mathcal{N}(\pi_1)$, which is equivalent to $e \circ \ell_n^{-1} \circ \cdots \circ \ell_1^{-1}$. By applying this rewriting top-down, we conclude that $\mathcal{N}(\pi_1) \preceq_{\text{path}} \mathcal{N}(^{-1})$. □

Hence, $\mathcal{N}(\pi)$ and $\mathcal{N}(^{-1})$ are Boolean-equivalent in expressive power.

7 Related work

Results on node-labeled trees are usually straightforward to translate to the edge-labeled trees we use. Benedikt et al. [3] studied path-equivalence of $\mathcal{N}(^{-1}, \pi)$ and its fragments on labeled trees:

Proposition 29 ([3, Proposition 2.1]). *For* $\mathcal{F}_1, \mathcal{F}_2 \subseteq \{^{-1}, \pi\}$, $\mathcal{N}(\mathcal{F}_1) \preceq_{\text{path}} \mathcal{N}(\mathcal{F}_2) \iff \mathcal{F}_1 \subseteq \mathcal{F}_2$.

Where applicable, we generalize Proposition 29 to Boolean separation, in Section 5. Wu et al. [26] proved a collapse result for relation algebra fragments with intersection:[12]

Proposition 30 ([26, Theorem 4.1]). *Both* $\mathcal{N}(^{-1}, \pi, \cap) \preceq_{\text{path}} \mathcal{N}(^{-1}, \pi)$ *and* $\mathcal{N}(^{-1}, \pi, \cap) \preceq_{\text{path}} \mathcal{N}(^{-1}, \cap)$.

In Section 6, we generalize Proposition 30 to also include coprojections.

Finally, Hellings et al. [16,17] studied the relative expressiveness of the fragments of $\mathcal{N}(\pi, \overline{\pi}, \cap, -)$,[13] and obtained the following results which are used in this study:

Proposition 31 ([16, Theorem 3]). *For* $\mathcal{F} \subseteq \{\pi, \overline{\pi}, \cap, -\}$, $\mathcal{N}(\mathcal{F}) \preceq_{\text{path}} \mathcal{N}(\overline{\mathcal{F}} - \{\cap, -\})$.

Proposition 32 ([17, Proposition 10]). *On unlabeled chains,* $\mathcal{N}(\text{di}) \not\preceq_{\text{path}} \mathcal{N}(\pi, \overline{\pi}, \cap, -)$ *and* $\mathcal{N}(^{-1}) \not\preceq_{\text{path}} \mathcal{N}(\pi, \overline{\pi}, \cap, -)$.

[11] Recall that the tree queries in Wu et al. [26] are essentially the same as the condition-free condition tree queries in this paper.

[12] Strictly speaking, they deal with union-free expressions, but since unions can always be pushed out to the outermost level, this is not a real restriction.

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

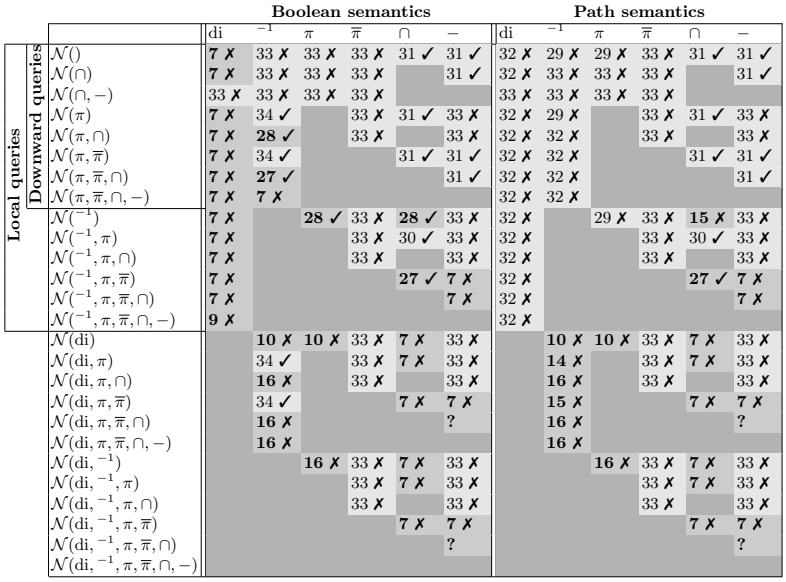

Fig. 12. Index to the separation and collapse results discussed in this paper. Let $(\mathcal{N}(\mathcal{F}), op)$ be a field in the "z semantics" part of the table, $z \in \{\text{bool}, \text{path}\}$. A check mark ✓ in the field $(\mathcal{N}(\mathcal{F}), op)$ means that $\mathcal{N}(\mathcal{F} \cup \{op\}) \preceq_z \mathcal{N}(\mathcal{F})$. A cross ✗ in the field $(\mathcal{N}(\mathcal{F}), op)$ means that $\mathcal{N}(\mathcal{F} \cup \{op\}) \not\preceq_z \mathcal{N}(\mathcal{F})$. Finally, a question mark **?** indicates an open problem.

Proposition 33 ([17, Proposition 19 and 21]). *We have* $\mathcal{N}(\pi) \not\preceq_{\text{bool}} \mathcal{N}()$, $\mathcal{N}(^{-1}) \not\preceq_{\text{bool}} \mathcal{N}()$, *and* $\mathcal{N}(\overline{\pi}) \not\preceq_{\text{bool}} \mathcal{N}(\text{di}, ^{-1}, \pi, \cap)$.

The graph query results of Fletcher et al. [10,11] include many separation results of which the proofs do not specialize to trees. They also proved a collapse result, that automatically does hold on trees:

Proposition 34 ([11, Proposition 4.2]). *Let* $\mathcal{F} \subseteq \{\text{di}, ^{-1}, \pi, \overline{\pi}\}$. *On labeled and unlabeled graphs, we have* $\mathcal{N}(\mathcal{F} \cup \{^{-1}\}) \preceq_{\text{bool}} \mathcal{N}(\mathcal{F} \cup \{\pi\})$.

Several other well-known expressiveness results are known in the context of Conditional XPath and Navigational XPath [6,22,23], 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. We observe that on chains, no sibling relation exists. Hence, the separation results we have proved on chains translate to separation results in the sibling-ordered tree data model.

8 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 12. 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 35. Let $\{\mathrm{di}, \overline{\pi}, \cap\} \subseteq \mathcal{F} \subseteq \{\mathrm{di}, ^{-1}, \pi, \overline{\pi}, \cap\}$ and let $z \in \{\mathrm{bool}, \mathrm{path}\}$. Do we have $\mathcal{N}(\mathcal{F} \cup \{-\}) \preceq_z \mathcal{N}(\mathcal{F})$ 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 several instances of expressions for which one can express the complement. For $k > 0$, we have, e.g., $\overline{\mathcal{E}^{-k}} \equiv_{\mathrm{path}} (\mathcal{E}^{-k} \circ \mathrm{di}) \cup (\overline{\pi}_1[\mathcal{E}^{-k}] \circ \mathrm{all})$. Unfortunately, we have not been able yet to express complement in every instance or been able to prove that expressing the complement is impossible in some instances. Additional difficulties arise from the fact that we can prove 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.

Another interesting direction for future work is augmenting the relation algebra with operators beyond the expressive power of FO[3]. Possible candidates would be an iteration construct such as an ancestor-descendant axis, or the more general and powerful Kleene-star transitive closure operator.

References

1. Barceló, P.: Querying graph databases. In: Proc. 32nd Symp. on Principles of Database Systems. pp. 175–188. ACM (2013)
2. Barceló, P., Pérez, J., Reutter, J.L.: Relative expressiveness of nested regular expressions. In: Proc. 6th A. Mendelzon Int'l Workshop on Foundations of Data Management. pp. 180–195. CEUR Workshop Proceedings (2012)
3. Benedikt, M., Fan, W., Kuper, G.: Structural properties of XPath fragments. Theor. Comput. Sci. 336(1), 3–31 (2005)
4. Benedikt, M., Koch, C.: XPath leashed. ACM Comput. Surv. 41(1), 3:1–3:54 (2009)
5. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F., Cowan, J.: Extensible markup language (XML) 1.1 (Second Edition). W3C Recommendation, W3C (2006), http://www.w3.org/TR/2006/REC-xml11-20060816
6. ten Cate, B.: The expressivity of XPath with transitive closure. In: Proc. 25th Symp. on Principles of Database Systems. pp. 328–337. ACM (2006)
7. Clark, J., DeRose, S.: XML Path Language (XPath) Version 1.0. W3C Recommendation, W3C (1999), http://www.w3.org/TR/1999/REC-xpath-19991116
8. Cruz, I.F., Mendelzon, A.O., Wood, P.T.: A graphical query language supporting recursion. In: Proc. 1987 ACM SIGMOD Int'l Conf. on Management of Data. pp. 323–330. ACM, New York, NY, USA (1987)
9. Ecma International: The JSON data interchange format, 1st Edition (2013), http://www.ecma-international.org/publications/standards/Ecma-404.htm

10. Fletcher, G.H.L., Gyssens, M., Leinders, D., Van den Bussche, J., Van Gucht, D., Vansummeren, S., Wu, Y.: Relative expressive power of navigational querying on graphs. In: Proc. 14th Int'l Conf. on Database Theory. pp. 197–207. ACM (2011)
11. Fletcher, G.H.L., Gyssens, M., Leinders, D., Surinx, D., Van den Bussche, J., Van Gucht, D., Vansummeren, S., Wu, Y.: Relative expressive power of navigational querying on graphs. Inform. Sci. 298, 390–406 (2015)
12. Givant, S.: The calculus of relations as a foundation for mathematics. J. Autom. Reason. 37(4), 277–322 (2006)
13. Grädel, E., Otto, M.: On logics with two variables. Theor. Comput. Sci. 224(1–2), 73–113 (1999)
14. Grohe, M.: Finite variable logics in descriptive complexity theory. The Bulletin of Symbolic Logic 4, 345–398 (1998)
15. Hellings, J.: 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)
16. Hellings, J., Gyssens, M., Wu, Y., Van Gucht, D., Van den Bussche, J., Vansummeren, S., Fletcher, G.H.L.: Relative expressive power of downward fragments of navigational query languages on trees and chains. In: Proc. 15th Symp. on Database Programming Languages. pp. 59–68 (2015)
17. Hellings, J., Gyssens, M., Wu, Y., Van Gucht, D., Van den Bussche, J., Vansummeren, S., Fletcher, G.H.L.: Comparing downward fragments of the relational calculus with transitive closure on trees. Tech. Rep. arXiv. Hasselt Univ. (2018), https://arxiv.org/abs/1803.01390
18. Hellings, J., Pilachowski, C.L., Van Gucht, D., Gyssens, M., Wu, Y.: From relation algebra to semi-join algebra: An approach for graph query optimization. In: Proc. 16th Int'l Symp. on Database Programming Languages. pp. 5:1–5:10 (2017)
19. Hidders, J., Paredaens, J., Van den Bussche, J.: J-logic: Logical foundations for JSON querying. In: Proc. 36th Symp. on Principles of Database Systems. pp. 137–149 (2017)
20. Kaushik, R., Bohannon, P., Naughton, J.F., Korth, H.F.: Covering indexes for branching path queries. In: Proc. 2002 ACM SIGMOD Int'l Conf. on Management of Data. pp. 133–144. ACM (2002)
21. Libkin, L.: Elements of Finite Model Theory. Springer Berlin Heidelberg (2004)
22. Marx, M.: Conditional XPath. ACM Trans. Database Syst. 30(4), 929–959 (2005)
23. Marx, M., de Rijke, M.: Semantic characterizations of navigational XPath. SIGMOD Rec. 34(2), 41–46 (2005)
24. Tarski, A.: On the calculus of relations. J. Symb. Log. 6(3), 73–89 (1941)
25. Tsichritzis, D.C., Lochovsky, F.H.: Hierarchical data-base management: a survey. ACM Comput. Surv. 8(1), 105–123 (1976)
26. Wu, Y., Van Gucht, D., Gyssens, M., Paredaens, J.: A study of a positive fragment of path queries: Expressiveness, normal form and minimization. Comput. J. 54(7), 1091–1118 (2011)