# Conjunctive Queries over Trees

GEORG GOTTLOB
Oxford University, Oxford, United Kingdom
CHRISTOPH KOCH
Universität des Saarlandes, Saarbrücken, Germany
KLAUS U. SCHULZ
Ludwig-Maximilians-Universität München, Munich, Germany

We study the complexity and expressive power of conjunctive queries over unranked labeled trees represented using a variety of structure relations such as "child", "descendant", and "following" as well as unary relations for node labels. We establish a framework for characterizing structures representing trees for which conjunctive queries can be evaluated efficiently. Then we completely chart the tractability frontier of the problem and establish a dichotomy theorem for our axis relations, i.e., we find all subset-maximal sets of axes for which query evaluation is in polynomial time and show that for all other cases, query evaluation is NP-complete. All polynomial-time results are obtained immediately using the proof techniques from our framework. Finally, we study the expressiveness of conjunctive queries over trees and show that for each conjunctive query, there is an equivalent acyclic positive query (i.e., a set of acyclic conjunctive queries), but that in general this query is not of polynomial size.

Categories and Subject Descriptors: E.1 [**DATA STRUCTURES**]: Trees; F.1.3 [**COMPUTATION BY ABSTRACT DEVICES**]: Complexity Classes – Reducibility and completeness; F.2.2 [**ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY**]: Nonnumerical Algorithms and Problems – Computations on discrete structures; H.2.3 [**DATABASE MANAGEMENT**]: Languages – Query languages; H.2.4 [**DATABASE MANAGEMENT**]: Systems – Query processing; I.7.2 [**TEXT PROCESSING**]: Document Preparation – Markup languages

General Terms: Theory, Languages, Algorithms

Additional Key Words and Phrases: Complexity, Expressiveness, Succinctness, Conjunctive queries, Trees, XML

## 1. INTRODUCTION

The theory of conjunctive queries over relational structures is, from a certain point of view, the greatest success story of the theory of database queries. These queries correspond to the most common queries in database practice, e.g. SQL select-from-where queries with conditions combined using "and" only. Their evaluation problem has also been considered in different contexts and under different names, notably as

---

This work was partially supported by project No. Z29-N04 of the Austrian Science Fund (FWF), by a project of the German Research Foundation (DFG), and by the REWERSE Network of Excellence of the European Union.
An extended abstract [Gottlob et al. 2004] of this work appeared in *Proc. 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2004)*, Paris, France, ACM Press, New York, USA, pp. 189 – 200.
Contact details: Georg Gottlob (Georg.Gottlob@comlab.ox.ac.uk), Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, United Kingdom.
Christoph Koch (koch@infosys.uni-sb.de), Lehrstuhl für Informationssysteme, Universität des Saarlandes, D-66123 Saarbrücken, Germany.
Klaus U. Schulz (schulz@cis.uni-muenchen.de), Centrum für Informations- und Sprachverarbeitung, Ludwig-Maximilians-Universität München, D-80536 München, Germany.

the *Constraint Satisfaction problem* in AI [Kolaitis and Vardi 1998; Dechter 2003] and the *H-coloring problem* in graph theory [Hell and Nesetril 2004]. Conjunctive queries are surprisingly well-behaved: Many important properties hold for conjunctive queries but fail for more general query languages (cf. [Chandra and Merlin 1977; Abiteboul et al. 1995; Maier 1983]).

Unranked labeled trees are a clean abstraction of HTML, XML, LDAP, and linguistic parse trees. This motivates the study of conjunctive queries *over trees*, where the tree structures are represented using unary node label relations and binary relations (often referred to as *axes*) such as *Child*, *Descendant*, and *Following*.

**XML Queries.** Conjunctive queries over trees are naturally related to the problem of evaluating queries (e.g., XQuery or XSLT) on XML data (cf. [Deutsch and Tannen 2003a]). However, conjunctive queries are a cleaner and simpler model whose complexity and expressiveness can be formally studied (while XQuery and XSLT are Turing-complete).

(Acyclic) conjunctive queries over trees are a generalization of the most frequently used fragment of XPath. For example, the XPath query //A[B]/following::C is equivalent to the (acyclic) conjunctive query

$$Q(z) \leftarrow A(x),\ \mathit{Child}(x, y),\ B(y),\ \mathit{Following}(x, z),\ C(z).$$

While XPath has been studied extensively (see e.g. [Gottlob et al. 2005; Gottlob et al. 2005] on its complexity, [Benedikt et al. 2003; Olteanu et al. 2002] on its expressive power, and [Hidders 2003] on the satisfiability problem), little work so far has addressed the theoretical properties of *cyclic* conjunctive queries over trees. Sporadic results on their complexity can be found in [Meuss et al. 2001; Gottlob and Koch 2002; 2004; Meuss and Schulz 2001].

**Data extraction and integration.** (Cyclic) conjunctive queries on trees have been used previously in data integration, where queries in languages such as XQuery were canonically mapped to conjunctive queries over trees to build upon the existing work on data integration with conjunctive queries [Deutsch and Tannen 2003a; 2003b]. Another application is Web information extraction using a datalog-like language over trees [Baumgartner et al. 2001; Gottlob and Koch 2004]. (Of course, each nonrecursive datalog rule is a conjunctive query.)

**Queries in computational linguistics.** A further area in which such queries are employed is computational linguistics, where one needs to search in, or check properties of, large corpora of parsed natural language. Corpora such as Penn Treebank [LDC 1999] are unranked trees labeled with the phrase structure of parsed (for Treebank, financial news) text. A query asking for prepositional phrases following noun phrases in the same sentence can be phrased as the conjunctive query

$$Q(z) \leftarrow S(x),\ \mathit{Descendant}(x, y),\ \mathit{NP}(y),\ \mathit{Descendant}(x, z),\ \mathit{PP}(z),\ \mathit{Following}(y, z).$$

Figure 1 shows this query in the intuitive graphical notation that we will use throughout the article (in which nodes correspond to variables, node labels to unary atoms, and edges to binary atoms).

**Dominance constraints.** Another important issue in computational linguistics are conjunctions of *dominance constraints* [Marcus et al. 1983], which turn out to be equivalent to (Boolean) conjunctive queries over trees. Dominance constraints

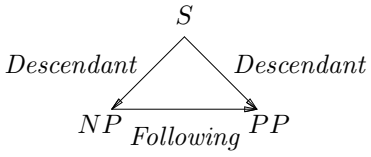



Fig. 1. A query graph.

have been influential as a means of incompletely specifying parse trees of natural language, in cases where (intermediate) results of parsing and disambiguation remain ambiguous. One problem of practical importance is the rewriting of sets of dominance constraints into equivalent but simpler sets (in particular, so-called *solved forms* [Bodirsky et al. 2004], which correspond to acyclic queries). This implies that studying the expressive power of conjunctive queries over trees, and the problem of deciding whether there is a set of acyclic conjunctive queries equivalent to a given conjunctive query, is relevant to computational linguistics.

**Higher-order unification.** The query evaluation problem for conjunctive queries over trees is also closely related to the *context matching problem*[1], a variant of the well-known context-unification problem [Schmidt-Schauß and Schulz 1998; 2002]. Some tractability frontier for the context matching problem is outlined in [Schmidt-Schauß and Stuber 2001]. However, little insight is gained from this for the database context, since the classes studied in [Schmidt-Schauß and Stuber 2001] become unnatural when formulated as conjunctive queries[2].

### Contributions

Given the substantial number of applications that we have hinted at above and the nice connection between database theory, computational linguistics, and term rewriting, it is surprising that conjunctive queries over trees have never been the object of a concerted study[3].

In particular, three questions seem worth studying:

(1) The complexity of (cyclic) conjunctive queries on trees has only been scratched in the literature. There is little understanding of how the complexity of conjunctive queries over trees depends on the relations used to model the tree.

(2) There is a natural connection between conjunctive queries and XPath. Since all XPath queries are acyclic, the question arises whether the acyclic positive queries (i.e., unions of acyclic conjunctive queries) are as expressive as the full class of conjunctive queries over trees.[4]

[1] To be precise, the analogy is most direct with ranked trees.

[2] These conjunctive queries require node inequality $\neq$ as a binary relation in addition to the tree structure relations. If $\neq$ is removed, the queries become acyclic. However, it is easy to see that already conjunctive queries using only the inequality relation over a fixed tree of three nodes are NP-complete, by a reduction from Graph 3-Colorability.

[3] Of course, as mentioned above, there are a number of papers that implicitly contain relevant results [Meuss et al. 2001; Meuss and Schulz 2001; Hidders 2003; Schmidt-Schauß and Stuber 2001]. The papers [Hell et al. 1996a; 1996b] address the complexity of a notion of tree homomorphisms that is uncomparable to the one used in database theory, and the results there are orthogonal.

[4] This is equivalent to asking whether for all conjunctive queries over trees there exist equivalent *positive Core XPath* queries [Gottlob et al. 2005].

(3) If that is the case, how much bigger do the acyclic versions of queries get than their cyclic counterparts? Except from being of theoretical interest, first translating queries into their acyclic versions, if that is possible, and then evaluating them as such may be a practical query evaluation strategy, because there are particularly good algorithms for evaluating such queries [Yannakakis 1981; Chekuri and Rajaraman 1997; Flum et al. 2002; Gottlob and Koch 2004].

We thus study conjunctive queries on tree structures represented using the XPath *axis* relations *child*, *descendant*, *descendant-or-self*, *following-sibling*, and *following*. Since we are free to use these relations with any pair of variables of our conjunctive queries (differently from XPath), these five axes render all others, i.e. *parent*, *ancestor*, *ancestor-or-self*, *preceding-sibling*, and *preceding*, redundant. Typed child axes such as *attribute* are redundant with the *child* axis and unary relations in our framework.

For a more elegant framework, we study the axes *Child*, $Child^*$ (= *descendant-or-self*), $Child^+$ (= *descendant*), *NextSibling*, $NextSibling^*$, $NextSibling^+$ (= *following-sibling*), and *Following*. (*NextSibling* and $NextSibling^*$ are not supported in XPath but are nevertheless considered here.) Subsequently, we denote this set of all axes considered in this article by $\mathbf{Ax}$.

The main contributions of this article are as follows.

—In [Gutjahr et al. 1992] it was shown that the *H-coloring problem* (cf. [Hell and Nesetril 2004]), and thus Boolean conjunctive query evaluation, on directed graphs $H$ that have the so-called *$\underline{X}$-property* (pronounced "X-underbar-property") is polynomial-time solvable. We determine which of our axis relations have the $\underline{X}$-property with respect to which orders of the domain elements. We show that the subset-maximal sets of axis relations for which the $\underline{X}$-property yields tractable query evaluation are the three disjoint sets

$$\{Child, NextSibling, NextSibling^*, NextSibling^+\},$$

$$\{Child^*, Child^+\}, \quad \text{and} \quad \{Following\}.$$

—We prove that the conjunctive query evaluation problem for queries involving any two axes that do not have the $\underline{X}$-property with respect to the same ordering of the tree nodes is NP-complete.
Thus the $\underline{X}$-property yields a complete characterization of the tractability frontier of the problem (under the assumption that P $\neq$ NP).

THEOREM 1.1. *Unless $P = NP$, for any $F \subseteq \mathbf{Ax}$, the conjunctive queries over structures with unary relations and binary relations from $F$ are in $P$ if and only if there is a total order $<$ such that all binary relations in $F$ have the $\underline{X}$-property w.r.t. $<$.*

Moreover we have the dichotomy that for any of our tree structures, the conjunctive query evaluation problem is either in P or NP-complete.
Table I shows the complexities of conjunctive queries over structures containing unary relations and either one or two axes.[5]

[5]It was shown in [Gottlob and Koch 2004] that conjunctive queries over *Child* and *NextSibling* are in P. Proposition 4.2 is from [Gottlob and Koch 2002].) The other results are new.

| | *Child* | *Child*$^+$ | *Child*$^*$ | *NextSibling* | *NextSibling*$^+$ | *NextSibling*$^*$ | *Following* |
|---|---|---|---|---|---|---|---|
| *Child* | in P (4.4) | NP-hard (5.1) | NP-hard (5.1) | in P (4.4) | in P (4.4) | in P (4.4) | NP-hard (5.2) |
| *Child*$^+$ | | in P (4.2) | in P (4.2) | NP-hard (5.7) | NP-hard (5.7) | NP-hard (5.7) | NP-hard (5.3) |
| *Child*$^*$ | | | in P (4.2) | NP-hard (5.5) | NP-hard (5.4) | NP-hard (5.6) | NP-hard (5.3) |
| *NextSibling* | | | | in P (4.4) | in P (4.4) | in P (4.4) | NP-hard (5.8) |
| *NextSibling*$^+$ | | | | | in P (4.4) | in P (4.4) | NP-hard (5.8) |
| *NextSibling*$^*$ | | | | | | in P (4.4) | NP-hard (5.8) |
| *Following* | | | | | | | in P (4.3) |

Table I. Complexity results for signatures with one or two axes, with pointers to relevant theorems.

All NP-hardness results hold already for fixed data trees (query complexity [Vardi 1982]). The polynomial-time upper bounds are established under the assumption that both data and query are variable (combined complexity).

—We study the expressive power of conjunctive queries on trees. We show that for each conjunctive query over trees, there is an equivalent acyclic positive query (APQ) over the same tree relations. The blowup in size of the APQs produced is exponential in the worst case.
It follows that there is an equivalent XPath query for each conjunctive query over trees, since each APQ can be translated into XPath (even in linear time).

—Finally, we provide a result that sheds some light at the succinctness of (cyclic) conjunctive queries and which demonstrates that the blow-up observed in our translation is actually necessary. We prove that there are conjunctive queries over trees for which no equivalent polynomially-sized APQ exists.

The structure of the article is as follows. We start with basic notions in Section 2. Section 3 introduces the $\underline{\mathrm{X}}$-property and the associated framework for finding classes of conjunctive queries that can be evaluated in polynomial time. Section 4 contains our polynomial-time complexity results. Section 5 completes our tractability frontier with the NP-hardness results. In Section 6, we provide our expressiveness results. Finally, we present our succinctness result in Section 7.

## 2. PRELIMINARIES

Let $\Sigma$ be a labeling alphabet. Throughout the article, if not explicitly stated otherwise, we will not assume $\Sigma$ to be fixed. An unranked tree is a tree in which each node may have an unbounded number of children. We allow for tree nodes to be labeled with multiple labels. However, throughout the article, our tractability results will support multiple labels while our NP-hardness and expressiveness results will not make use of them.

We represent trees as relational structures using unary label relations $(\mathit{Label}_a)_{a\in\Sigma}$ and binary relations called axes. For a relational structure $\mathcal{A}$, let $A = |\mathcal{A}|$ denote the finite domain (in the case of a tree, the nodes) and let $||\mathcal{A}||$ denote the size of the structure under any reasonable encoding scheme (see e.g. [Ebbinghaus and Flum 1999]). We use the binary *axis* relations *Child* (defined in the normal way) and *NextSibling* (where $\mathit{NextSibling}(v, w)$ if and only if $w$ is the right neighboring

sibling of $v$ in the tree), their transitive and reflexive and transitive closures (denoted $\mathit{Child}^+$, $\mathit{NextSibling}^+$, $\mathit{Child}^*$, $\mathit{NextSibling}^*$), and the axis *Following*, defined as

$$\mathit{Following}(x,y) = \exists z_1 \exists z_2\ \mathit{Child}^*(z_1,x) \wedge \mathit{NextSibling}^+(z_1,z_2) \wedge \mathit{Child}^*(z_2,y). \quad (1)$$

This set of axes covers the standard XPath axes (cf. [World Wide Web Consortium 1999]) by the equivalences $\mathit{Child}^+ = \mathit{Descendant}$, $\mathit{Child}^* = \mathit{Descendant\text{-}or\text{-}self}$, and $\mathit{NextSibling}^+ = \mathit{Following\text{-}sibling}$.

We consider three well-known total orderings on finite ordered trees. The *pre-order* $\leq_{pre}$ corresponds to a depth-first left-to-right traversal of a tree. If XML-documents are represented as trees in the usual way, the pre-order coincides with the *document order*. It is given by the sequence of *opening* tags of the XML *elements* (corresponding to nodes). The *post-order* $\leq_{post}$ corresponds to a bottom-up left-to-right traversal of the tree and is given by the sequence of *closing* tags of elements. Furthermore, we also consider the ordering $\leq_{bflr}$ which is given by the sequence of opening tags if we traverse the tree breadth-first left-to-right.

The $k$-ary *conjunctive queries* can be defined by positive existential first-order formulas without disjunction and with $k$ free variables. We will usually use the standard (datalog) *rule notation* for conjunctive queries (cf. [Abiteboul et al. 1995]).

We call the 0-ary queries *Boolean* and the unary queries *monadic*. The *containment* of queries $Q$ and $Q'$ is defined in the normal way: Query $Q$ is said to be contained in $Q'$ (denoted $Q \subseteq Q'$) iff, for all tree structures $\mathcal{A}$, $Q'$ returns at least all tuples on $\mathcal{A}$ that $Q$ returns on $\mathcal{A}$. (To cover Boolean queries, tuples here may be nullary.) Two queries $Q, Q'$ are called equivalent iff $Q \subseteq Q'$ and $Q' \subseteq Q$.

Let $Q$ be a conjunctive query and let $\mathit{Var}(Q)$ denote the variables appearing in $Q$. The query graph of $Q$ over unary and binary relations is the *directed* multigraph $G = (V, E)$ with edge labels and multiple node labels such that $V = \mathit{Var}(Q)$, node $x$ is labeled $P$ iff $Q$ contains unary atom $P(x)$, and $E$ contains labeled directed edge $x \xrightarrow{R} y$ if and only if $Q$ contains binary atom $R(x,y)$. Figure 1 shows an example of such a query graph. Our notion of query graph is sometimes called *positive atomic diagram* in model theory or the graph of the *canonical database* of a query in the database theory literature.

Throughout the article, we use lower case node and variable names and upper case label and relation names.

## 3. THE X-PROPERTY

Let $Q$ be a conjunctive query and let $A$ denote the finite domain, i.e. in case of a tree the set of nodes. A pre-valuation for $Q$ is a total function $\Theta : \mathit{Var}(Q) \to 2^A$ that assigns to each variable of $Q$ a *nonempty* subset of $A$. A valuation for $Q$ is a total function $\theta : \mathit{Var}(Q) \to A$.

Let $\mathcal{A}$ be a relational structure of unary and binary relations. A pre-valuation $\Theta$ is called *arc-consistent*[6] iff for each unary atom $P(x)$ in $Q$ and each $v \in \Theta(x)$, $P(v)$ is true (in $\mathcal{A}$) and for each binary atom $R(x,y)$ in $Q$, for each $v \in \Theta(x)$ there exists $w \in \Theta(y)$ such that $R(v,w)$ is true and for each $w \in \Theta(y)$ there exists $v \in \Theta(x)$ such that $R(v,w)$ is true.

[6]This notion is well-known in constraint satisfaction, cf. [Dechter 2003].

PROPOSITION 3.1 (FOLKLORE). *There is an algorithm which checks in time* $O(||\mathcal{A}|| \cdot |Q|)$ *whether an arc-consistent pre-valuation of* $Q$ *on* $\mathcal{A}$ *exists, and if it does, returns one.*

PROOF. We phrase the problem of computing $\Theta$ by deciding, for each $x, v$, whether $v \notin \Theta(x)$ as an instance $\mathcal{P}$ of propositional Horn-SAT. The propositional predicates are the atoms $\mathit{Remove}(x, v)$ (where $x \in \mathit{Vars}(Q)$, $v \in A$ are constants), and the Horn clauses are

$$\{\mathit{Remove}(x,v) \leftarrow . \mid P(x) \in Q,\ v \in A,\ \neg P^{\mathcal{A}}(v)\} \cup$$
$$\{\mathit{Remove}(x,v) \leftarrow \bigwedge\{\mathit{Remove}(y,w) \mid R^{\mathcal{A}}(v,w)\}. \mid R(x,y) \in Q,\ v \in A\} \cup$$
$$\{\mathit{Remove}(y,w) \leftarrow \bigwedge\{\mathit{Remove}(x,v) \mid R^{\mathcal{A}}(v,w)\}. \mid R(x,y) \in Q,\ w \in A\}$$

Let *Remove* be the binary relation defined by $\mathcal{P}$ and let

$$T = (\mathit{Vars}(Q) \times A) - \mathit{Remove}$$

be the complement of that relation. If there is a variable $x$ such that for no node $v$, $(x, v) \in T$, no arc-consistent pre-valuation of $Q$ on $\mathcal{A}$ exists and $Q$ is not satisfied. Otherwise, the pre-valuation defined by

$$\Theta(x) \mapsto \{v \mid (x, v) \in T\},$$

for each $x$, is obviously arc-consistent and contains all arc-consistent pre-valuations of $Q$ and $\mathcal{A}$.

Program $\mathcal{P}$ can be computed and solved (e.g. using Minoux' algorithm [Minoux 1988]), and the solution complemented, in time linear in the size of the program, which is $O(||\mathcal{A}|| \cdot |Q|)$. □

Actually, this algorithm computes the unique subset-maximal arc-consistent pre-valuation of $Q$ on $\mathcal{A}$.

A valuation $\theta$ is called *consistent* if it satisfies the query. In this case, for a Boolean query, we also say that the structure is a *model* of the query and the valuation a *satisfaction*. Obviously, a valuation is consistent if and only if the pre-valuation $\Theta$ defined by $\Theta(x) \mapsto \{\theta(x)\}$ is arc-consistent. Let $<$ be a total order on $A = |\mathcal{A}|$ and $\Theta$ be a pre-valuation. Then the valuation $\theta$ with $\theta(x) \mapsto v$ iff $v$ is the smallest node in $\Theta(x)$ w.r.t. $<$ is called the *minimum valuation* w.r.t. $<$ in $\Theta$.

DEFINITION 3.2. Let $\mathcal{A}$ be a relational structure, $R$ a binary relation in $\mathcal{A}$, and $<$ a total order on $A = |\mathcal{A}|$. Then, $R$ is said to have the *$\underline{\text{X}}$-property w.r.t.* $<$ iff for all $n_0, n_1, n_2, n_3 \in A$ such that $n_0 < n_1$ and $n_2 < n_3$,

$$R(n_1, n_2) \wedge R(n_0, n_3) \Rightarrow R(n_0, n_2).$$

Figure 2 illustrates why the property is called $\underline{\text{X}}$ (read as "X-underbar"). Let us consider two vertical bars both representing the order $<$ bottom-up (i.e., with the smallest value at the bottom). Let each edge $(u, v)$ in $R$ be represented by an arc from node $u$ on the left bar to node $v$ on the right bar. Then, whenever there are two crossing arcs $(u, v)$ and $(u', v')$ in this diagram, then there must be an arc $(\min(u, u'), \min(v, v'))$, the "underbar", in the diagram as well.

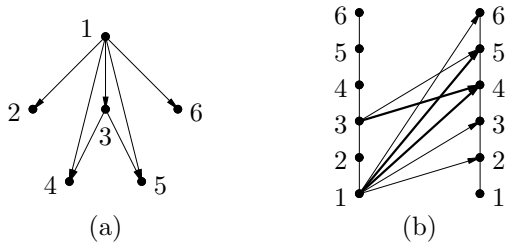



Fig. 2. The X-property. Graph (a) and its illustration by arcs between two bars (b). For crossing arcs $R(u, v)$ and $R(u', v')$, say $u < u'$ and $v' < v$, there must be an arc $R(u, v')$.

REMARK 3.3. The X-property[7] was introduced in [Gutjahr et al. 1992], where it was shown that the $H$-coloring problem (or equivalently the conjunctive query evaluation problem) on graphs $H$ with the X-property is polynomial-time solvable (see also [Hell and Nesetril 2004]). In the remainder of this section, we rephrase this result as a tool for efficiently evaluating conjunctive queries.

Let $\mathcal{A}$ be a structure of unary and binary relations and let $<$ be a total order on $|\mathcal{A}|$. Structure $\mathcal{A}$ is said to have the X-property w.r.t. $<$ if all binary relations $R$ in $\mathcal{A}$ have the X-property w.r.t. $<$.

LEMMA 3.4. *Let $\mathcal{A}$ be a structure with the X-property w.r.t. $<$ and let $\Theta$ be an arc-consistent pre-valuation on $\mathcal{A}$ for a given conjunctive query over the relations of $\mathcal{A}$. Then, the minimum valuation in $\Theta$ w.r.t. $<$ is consistent.*

PROOF. Let $\theta$ denote the minimum valuation in $\Theta$ w.r.t. $<$. To prove $\theta$ consistent, we only need to show the following: If $R(x, y)$ is any binary atom of $Q$ with variables $x, y$ then $R(x, y)$ holds under assignment $\theta$, i.e. $R(\theta(x), \theta(y))$ is true in $\mathcal{A}$.

Let $\theta(x) = n_0$ and $\theta(y) = n_2$. Since $\Theta$ is arc-consistent there exists a node $n_1 \in \Theta(x)$ such that $R(n_1, n_2)$ and a node $n_3 \in \Theta(y)$ such that $R(n_0, n_3)$. If $n_0 = n_1$ or $n_2 = n_3$ then $R(\theta(x), \theta(y)) = R(n_0, n_2)$ is true and we are done. Otherwise, since $\theta$ is a minimum valuation we have $n_0 < n_1$ (because $n_0 = \theta(x) = \min \Theta(x)$, $n_1 \in \Theta(x)$, and $n_0 \neq n_1$) and $n_2 < n_3$ (because $n_2 = \theta(y) = \min \Theta(y)$, $n_3 \in \Theta(y)$, and $n_2 \neq n_3$). Then it follows from Definition 3.2 that $R(n_0, n_2)$. □

Clearly, if no arc-consistent pre-valuation of $Q$ on $\mathcal{A}$ exists, there is no consistent valuation for $Q$ on $\mathcal{A}$.

THEOREM 3.5. *Given a structure $\mathcal{A}$ with the X-property w.r.t. $<$ and a Boolean conjunctive query $Q$ over $\mathcal{A}$, $Q$ can be evaluated on $\mathcal{A}$ in time $O(||\mathcal{A}|| \cdot |Q|)$.*

PROOF. By Lemma 3.4, all we need to do to check whether a Boolean query $Q$ is satisfied is to try to compute the subset-maximal arc-consistent pre-valuation $\Theta$ with respect to $Q$. By Proposition 3.1, this can be done in time $O(||\mathcal{A}|| \cdot |Q|)$. If it exists, $Q$ returns true; otherwise, $Q$ returns false. □

If follows that checking whether a given tuple $\langle a_1, \ldots, a_k \rangle$ is in the result of a $k$-ary conjunctive query on structures with the X-property w.r.t. some order

[7] In [Gottlob et al. 2004], this property was called *hemichordality*.

can be decided in time $O(||\mathcal{A}|| \cdot |Q|)$ as well. All we need to do is to add (new) singleton unary relations $X_1 = \{a_1\}, \ldots, X_k = \{a_k\}$ to $\mathcal{A}$ and to rewrite the query $Q(x_1, \ldots, x_k) \leftarrow \Phi(x_1, \ldots, x_k)$ into the Boolean query $Q \leftarrow \Phi(x_1, \ldots, x_k) \wedge X_1(x_1) \wedge \cdots \wedge X_k(x_k)$. A $k$-ary conjunctive query $Q$ over $\mathcal{A}$ with $A = |\mathcal{A}|$ can thus be evaluated on $\mathcal{A}$ in time $O(|A|^k \cdot ||\mathcal{A}|| \cdot |Q|)$.

For relations that are subsets of the given total order $\leq$ (the reflexive closure of $<$), a slightly stronger condition for the $\underline{\text{X}}$-property w.r.t. $<$ can be given.

LEMMA 3.6. *Let $\mathcal{A}$ be a structure, $<$ a total order on $A = |\mathcal{A}|$, and $R$ a binary relation of $\mathcal{A}$ such that $R \subseteq \leq$. Then, $R$ has the $\underline{X}$-property w.r.t. $<$ iff for all $n_0, n_1, n_2, n_3 \in A$ such that $n_0 < n_1 \leq n_2 < n_3$,*

$$R(n_1, n_2) \wedge R(n_0, n_3) \Rightarrow R(n_0, n_2).$$

PROOF. Obviously, if $R$ has the $\underline{\text{X}}$-property w.r.t. $<$, then the condition of Definition 3.2 implies that the condition of our lemma holds. Conversely, since for all $n_1, n_2$, $R(n_1, n_2) \Rightarrow n_1 \leq n_2$, by our lemma, for all $n_0, n_1, n_2, n_3 \in A$ such that $n_0 < n_1$ and $n_2 < n_3$, $R(n_1, n_2) \wedge R(n_0, n_3) \Rightarrow R(n_0, n_2)$. □

A symmetric version of Lemma 3.5 holds for relations $R \subseteq \geq$.

LEMMA 3.7. Let $\mathcal{A}$ be a structure, $<$ a total order on $A = |\mathcal{A}|$, and $R$ a binary relation of $\mathcal{A}$ such that $R \subseteq \geq$. Then, $R$ has the $\underline{\text{X}}$-property w.r.t. $<$ iff for all $n_0, n_1, n_2, n_3 \in A$ such that $n_0 < n_1 \leq n_2 < n_3$,

$$R(n_2, n_1) \wedge R(n_3, n_0) \Rightarrow R(n_2, n_0).$$

PROOF. Let $R' = R^{-1}$. By Lemma 3.6, $R'$ has the $\underline{\text{X}}$-property w.r.t. $<$ precisely if for all $n_0, n_1, n_2, n_3 \in A$, $n_0 < n_1 \leq n_2 < n_3 \wedge R'(n_1, n_2) \wedge R'(n_0, n_3) \Rightarrow R'(n_0, n_2)$. Thus, $R$ has the $\underline{\text{X}}$-property w.r.t. $<$ iff the condition of our lemma holds. □

## 4. POLYNOMIAL-TIME RESULTS

The results of Section 3 provide us with a simple technique for proving polynomial-time complexity results for conjunctive queries over trees. Indeed, there is a wealth of inclusions of axis relations in the total orders introduced in Section 2:

(1) all the axes in **Ax** are subsets of the pre-order $\leq_{pre}$,

(2) $\mathit{Child}^{-1}$, $(\mathit{Child}^+)^{-1}$, $(\mathit{Child}^*)^{-1}$, *Following*, *NextSibling*, $\mathit{NextSibling}^+$, and $\mathit{NextSibling}^*$ are subsets of the post-order $\leq_{post}$, and

(3) *Child*, $\mathit{Child}^+$, $\mathit{Child}^*$, *NextSibling*, $\mathit{NextSibling}^+$, and $\mathit{NextSibling}^*$ are subsets of the order $\leq_{bflr}$.

Using Lemma 3.6, it is straightforward to show that

THEOREM 4.1. *The axes*

(*1*) $\mathit{Child}^+$ *and* $\mathit{Child}^*$ *have the* $\underline{X}$*-property w.r.t.* $<_{pre}$,

(*2*) *Following has the* $\underline{X}$*-property w.r.t.* $<_{post}$, *and*

(*3*) *Child, NextSibling,* $\mathit{NextSibling}^*$, *and* $\mathit{NextSibling}^+$ *have the* $\underline{X}$*-property w.r.t.* $<_{bflr}$.

PROOF. All proof arguments use Lemma 3.6.

We first show that $\mathit{Child}^*$ has the $\underline{\mathrm{X}}$-property w.r.t. $<_{pre}$. (The proof for $\mathit{Child}^+$ is similar.) Consider the nodes $n_0, \ldots, n_3$ such that $n_0 <_{pre} n_1 \leq_{pre} n_2 <_{pre} n_3$, $\mathit{Child}^*(n_0, n_3)$, and $\mathit{Child}^*(n_1, n_2)$. It is simple to see that $\leq_{pre}$ is the disjoint union of $\mathit{Child}^*$ and $\mathit{Following}$. Therefore, either $\mathit{Child}^*(n_0, n_1)$, which implies $\mathit{Child}^*(n_0, n_2)$, or $\mathit{Following}(n_0, n_1)$. The latter case would yield $n_3 <_{pre} n_1$, a contradiction.

Next, we show that $\mathit{Following}$ has the $\underline{\mathrm{X}}$-property w.r.t. $<_{post}$. Assume that

$$n_0 <_{post} n_1 \leq_{post} n_2 <_{post} n_3$$

and $\mathit{Following}(n_1, n_2)$, $\mathit{Following}(n_0, n_3)$. Clearly, the relation $\leq_{post}$ is the disjoint union of $\mathit{Following}$ and the inverse of $\mathit{Child}^*$. Since $n_0 <_{post} n_1$ is true, either $\mathit{Child}^*(n_1, n_0)$ or $\mathit{Following}(n_0, n_1)$ must hold. In both cases it follows that $\mathit{Following}(n_0, n_2)$. Thus, $\mathit{Following}$ has the $\underline{\mathrm{X}}$-property w.r.t. $<_{post}$.

The fact that $\mathit{Child}$ has the $\underline{\mathrm{X}}$-property w.r.t. $<_{bflr}$ follows vacuously from the characterization of Lemma 3.6: Assume that $n_0 <_{bflr} n_1 \leq_{bflr} n_2 <_{bflr} n_3$ and that $\mathit{Child}(n_1, n_2)$ (thus $n_0 <_{bflr} n_1 <_{bflr} n_2 <_{bflr} n_3$) and $\mathit{Child}(n_0, n_3)$. Because of $\mathit{Child}(n_0, n_3)$, the node $n_1$ is at most one level below $n_0$ in the tree. There are two cases, (1) $\mathit{Following}(n_0, n_1)$ and $n_0$, $n_1$ are on the same level in the tree or (2) $\mathit{Following}(n_1, n_3)$ and $n_1$, $n_3$ are on the same level in the tree. In case (1), since $n_3$ is a child of $n_0$ and $n_2$ is a child of $n_1$, $n_3 <_{bflr} n_2$, contradiction. In case (2), since $n_2$ is a child of $n_1$, $n_2$ is one level below $n_3$ in the tree and thus $n_3 <_{bflr} n_2$, contradiction.

It is easy to verify that $\mathit{NextSibling}$, $\mathit{NextSibling}^*$, and $\mathit{NextSibling}^+$ have the $\underline{\mathrm{X}}$-property w.r.t. $<_{bflr}$ using Lemma 3.6. □

Now, it follows immediately from Theorem 3.5 that

COROLLARY 4.2 [GOTTLOB AND KOCH 2002]. *Conjunctive queries over*

$$\tau_1 := \langle (\mathit{Label}_a)_{a \in \Sigma}, \mathit{Child}^+, \mathit{Child}^* \rangle$$

*are in polynomial time w.r.t. combined complexity.*

COROLLARY 4.3. *Conjunctive queries over the signature*

$$\tau_2 := \langle (\mathit{Label}_a)_{a \in \Sigma}, \mathit{Following} \rangle$$

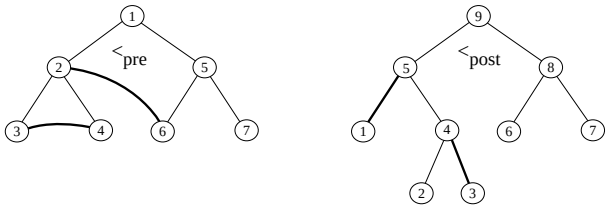
*are in polynomial time w.r.t. combined complexity.*

COROLLARY 4.4. *Conjunctive queries over the signature*

$$\tau_3 := \langle (\mathit{Label}_a)_{a \in \Sigma}, \mathit{Child}, \mathit{NextSibling}, \mathit{NextSibling}^*, \mathit{NextSibling}^+ \rangle$$

*are in polynomial time w.r.t. combined complexity.*

EXAMPLE 4.5. The remaining inclusions between axis relations and total orders introduced at the beginning of this section do not extend to the $\underline{\mathrm{X}}$-property. For example, Figure 3 (a) illustrates that $\mathit{Following}$ does not satisfy Lemma 3.6 w.r.t. pre-order $<_{pre}$. (While $2 <_{pre} 3 <_{pre} 4 <_{pre} 6$, $\mathit{Following}(2, 6)$ and $\mathit{Following}(3, 4)$ hold, $\mathit{Following}(2, 4)$ does not hold.)




(a) (b)

Fig. 3. (a) *Following* does not have the X-property w.r.t. $<_{pre}$; (b) $Descendant^{-1}$ and $Descendant\text{-}or\text{-}self^{-1}$ do not have the X-property w.r.t. $<_{post}$.

Figure 3 (b) shows that $Descendant^{-1}$ and $Descendant\text{-}or\text{-}self^{-1}$ do not satisfy the condition of Lemma 3.7 w.r.t. post-order $<_{post}$. (While $1<_{post}3<_{post}4<_{post}5$, $Descendant^{-1}(1,5)$ and $Descendant^{-1}(3,4)$ hold, $Descendant^{-1}(1,4)$ does not hold.)

For total order $<$, let $Succ_< := \{\langle x,y\rangle \mid x < y \ \wedge \nexists z\ x < z < y\}$. It is trivial to verify that $Succ_<$, $<$, and $\leq$ have the X-property w.r.t. $<$. Thus, we may for instance add the relations $<_{pre}$ (document order) and $Succ_{<_{pre}}$ ("next node in document order") to $\tau_1$, while retaining polynomial-time combined complexity.

## 5. NP-HARDNESS RESULTS

In this section, we study the complexity of the conjunctive query evaluation problem for the remaining sets of axis relations. For all cases for which our techniques based on the X-property do not yield a polynomial-time complexity result, we are able to prove NP-hardness. All NP-hardness results hold already for query complexity, i.e., in a setting where the data tree, and thus in particular the labeling alphabet, is fixed and only the query is assumed variable.

All reductions are from *one-in-three 3SAT*, which is the following NP-complete problem: Given a set $U$ of variables, a collection $\mathbf{C}$ of clauses over $U$ such that each clause $C \in \mathbf{C}$ has $|C| = 3$, is there a truth assignment for $U$ such that each clause in $\mathbf{C}$ has exactly one true literal? 1-in-3 3SAT remains NP-complete if all clauses contain only positive literals [Schaefer 1978].

Below, we will use shortcuts of the form $\chi^k(x,y)$, where $\chi$ is an axis, in queries to denote chains of $k$ $\chi$-atoms leading from variable $x$ to $y$. For example, $Child^2(x,y)$ is a shortcut for $Child(x,z), Child(z,y)$, where $z$ is a new variable.

The first theorem strengthens a known result for combined complexity [Meuss et al. 2001] to query complexity.

THEOREM 5.1. *Conjunctive queries over the signatures*

$$\tau_4 := \langle (Label_a)_{a\in\Sigma}, Child, Child^+\rangle$$
$$\tau_5 := \langle (Label_a)_{a\in\Sigma}, Child, Child^*\rangle$$

*are NP-complete w.r.t. query complexity.*

PROOF. Here, as in all other proofs of this section, we only need to show NP-hardness. Let $C_1,\ldots,C_m$ be a 1-in-3 3SAT instance with positive literals only. We assume that $C_i$ is an ordered sequence of three positive literals. We may assume without loss of generality that no clause contains a particular literal more than

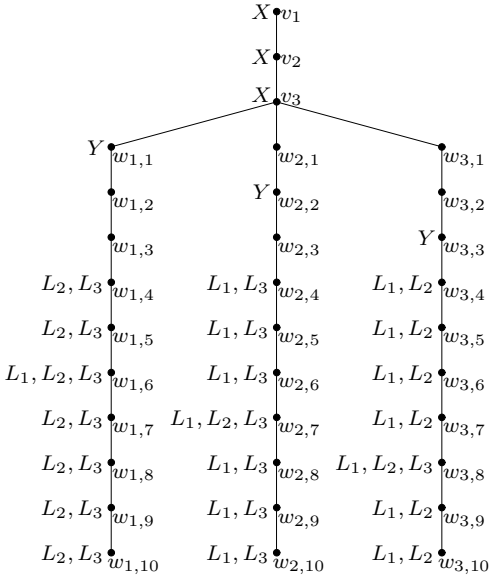



Fig. 4. Data tree of the proof of Theorem 5.1.

once. We reduce this instance to one of the Boolean conjunctive query evaluation problem for $\tau_4$ ($\tau_5$).

The fixed data tree over alphabet $\{X, Y, L_1, L_2, L_3\}$ is shown in Figure 4.

For the query, we introduce variables $x_i, y_i$ for $1 \leq i \leq m$ and in addition a variable $z_{k,l,i,j}$ whenever the $k$-th literal of $C_i$ coincides with the $l$-th literal of $C_j$ ($1 \leq i \leq m$, $1 \leq j \leq m$, $i \neq j$, $1 \leq k, l \leq 3$).

The Boolean query consists of the following atoms:

—for $1 \leq i \leq m$,

$$X(x_i),\ Y(y_i),\ \mathit{Child}^3(x_i, y_i),$$

—for each variable $z_{k,l,i,j}$,

$$L_k(z_{k,l,i,j}), \mathit{Child}^{\circ}(y_i, z_{k,l,i,j}), \mathit{Child}^{8+k-l}(x_j, z_{k,l,i,j})$$

where $\circ$ is "+" on signature $\tau_4$ and "$*$" on $\tau_5$.

"$\Rightarrow$". To prove correctness of the reduction, we first show that given any solution mapping $\sigma : \{1, \ldots, m\} \to \{1, 2, 3\}$ of $C_1, \ldots, C_m$ (i.e., $\sigma(i) = k'$ iff $\sigma$ selects the $k'$-th literal from $C_i$) we can define a satisfaction $\theta$ of the query. We first define a valuation $\theta$ of our query and then show that all query atoms are satisfied. We set

—$\theta(x_i) := v_{\sigma(i)}$ for $1 \leq i \leq m$,
—$\theta(y_i) := w_{\sigma(i),\sigma(i)}$ for $1 \leq i \leq m$, and
—for each variable $z_{k,l,i,j}$, $\theta(z_{k,l,i,j}) := w_{\sigma(i),5+k-l+\sigma(j)}$.

We now prove that $\theta$ is a satisfaction of the query. Our choice of $\theta$ implies that the variables $x_i$ and $y_i$ are mapped to nodes with labels $X$ and $Y$, respectively. Furthermore, $\theta(y_i) = w_{\sigma(i),\sigma(i)}$ can be reached from $\theta(x_i) = v_{\sigma(i)}$ with three child-steps. For any variable of the form $z_{k,l,i,j}$, $\theta(z_{k,l,i,j}) = w_{\sigma(i),5+k-l+\sigma(j)}$ is always

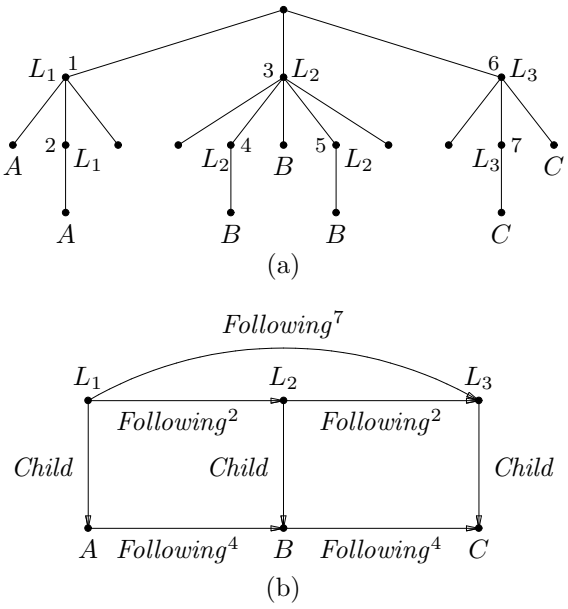



Fig. 5. Clause gadget of proof of Theorem 5.2.

a *Child*$^\circ$ of $w_{\sigma(i),\sigma(i)}$. If $\sigma(i) \neq k$, then $\theta(z_{k,l,i,j}) = w_{\sigma(i),5+k-l+\sigma(j)}$ has label $L_k$ because $4 \leq 5 + k - l + \sigma(j) \leq 10$ and the nodes $w_{\sigma(i),4}, \ldots, w_{\sigma(i),10}$ all have (at least) the two labels $L_{k'}$ for which $\sigma(i) \neq k'$. If $\sigma(i) = k$, then $\sigma(j) = l$. By going $8 + k - l$ steps downward from $v_{\sigma(j)}$, passing through $w_{k,k}$, we reach node $w_{k,5+k}$, which has label $L_k$. Since $\theta(z_{k,l,i,j}) = w_{\sigma(i),5+k-l+\sigma(j)} = w_{k,5+k}$, the query atoms $\mathit{Child}^{8+k-l}(x_j, z_{k,l,i,j})$ are satisfied. Therefore, $\theta$ is indeed a satisfaction of our query.

"$\Leftarrow$". To finish the proof we show that from any satisfaction $\theta$ of the query we obtain a corresponding solution for the 1-in-3 3SAT instance $C_1, \ldots, C_m$. If $\theta(x_i) = v_k$, we interpret this as the $k$-th literal of clause $C_i$ being chosen to be true. Obviously, under any valuation of the query, we select precisely one literal from each clause $C_i$. We have to verify that if a literal $L$ occurs in two clauses $C_i$ and $C_j$ and we select $L$ in $C_i$, we also select $L$ in $C_j$. Let $L$ be the $k$-th literal of $C_i$ and let $\theta(x_i) = v_k$ (i.e., $L$ is selected in $C_i$). Then $\theta(z_{k,l,i,j}) = w_{k,5+k}$ because that is the only node below $\theta(y_i) = w_{k,k}$ that has label $L_k$. The query contains the atom $\mathit{Child}^{8+k-l}(x_j, z_{k,l,i,j})$ for variable $z_{k,l,i,j}$. From node $w_{k,5+k}$, by $8 + k - l$ upward steps we arrive at node $v_l$. Hence $\theta(x_j) = v_l$, and we select $L$ from clause $C_j$.

Some nodes in the data tree carry multiple labels. However, since the *Child* axis is available in both $\tau_4$ and $\tau_5$, multiple labels can be eliminated by pushing them down to new children in the data tree and modifying the queries accordingly. □

THEOREM 5.2. *Conjunctive queries over the signature*

$$\tau_6 := \langle (\mathit{Label}_a)_{a \in \Sigma}, \mathit{Child}, \mathit{Following} \rangle$$

*are NP-complete w.r.t. query complexity.*

PROOF. Figure 5 shows (a) the fragment of a data tree and (b) a query over the labeling alphabet $\Sigma = \{A, B, C, L_1, L_2, L_3\}$.

Observe that the labels $L_1$, $L_2$, and $L_3$ occur only once each in Figure 5 (b). We will refer to the nodes (= query variables) labeled $L_1$, $L_2$, and $L_3$ by $v_1$, $v_2$, and $v_3$, respectively. For the following discussion, we have annotated some of the nodes of the data tree with numbers (1–7). Below, node 1 (resp. 3, 6) is called the *topmost position* of variable $v_1$ (resp. $v_2$, $v_3$). We start with two simple observations.

(1) *In any satisfaction $\theta$ of the query on the data tree, at most one of the variables $v_1$, $v_2$, and $v_3$ is mapped to its topmost position under $\theta$.* In fact, assume, e.g., that $\theta(v_1) = 1$. From node 1, node 3 (resp. 6) cannot be reached by a sequence of 2 (resp. 7) *Following*-steps. Hence we have $\theta(v_2) \neq 3$ and $\theta(v_3) \neq 6$.

(2) *In any solution $\theta$ of the problem, at least one of the variables $v_1$, $v_2$, and $v_3$ is mapped to its topmost position under $\theta$.* In fact, assume that $\theta(v_1) = 2$ and $\theta(v_2) \neq 3$. The atoms in the query (in particular, on the variables corresponding to nodes on the bottom of the query graph) require that $\theta(v_2) \neq 4$. Hence $\theta(v_2) = 5$ is the only remaining possibility. But now the query requires that $\theta(v_3) \neq 7$. Hence $\theta(v_3) = 6$.

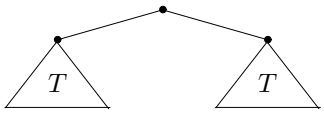
Thus, precisely the three partial assignments

$$(a)\ \ \theta(v_1) := 1,\ \theta(v_2) := 4,\ \theta(v_3) := 7$$
$$(b)\ \ \theta(v_1) := 2,\ \theta(v_2) := 3,\ \theta(v_3) := 7$$
$$(c)\ \ \theta(v_1) := 2,\ \theta(v_2) := 5,\ \theta(v_3) := 6$$

can be extended to a satisfaction of the query. Precisely one of the variables $v_1$, $v_2$, and $v_3$ is mapped to its topmost position under each of the above assignments. Conversely, for each variable there is a satisfying assignment in which it takes its topmost position.

Given a clause $C$, an *ordered* list of three positive literals, we interpret a satisfaction $\theta$ in which variable $v_k$ is mapped to its topmost position as the selection of the $k$-th literal from $C$ to be true. The encoding described above thus assures that exactly one variable of clause $C$ is selected and becomes true.

Now consider a 1-in-3 3SAT problem instance over positive literals with clauses $C_1, \ldots, C_m$. We encode such an instance as a conjunctive query over $\tau_6$ and a fixed data tree over labeling alphabet $\Sigma = \{A, B, C, L_1, L_2, L_3\}$. This tree consists of two copies of the tree of Figure 5 (a) under a common root, i.e.,




where $T$ denotes the tree of Figure 5 (a).

The query is obtained as follows. Each clause $C_i$ is represented using two copies of the query gadget of Figure 5 (b) (a "left" copy $Q_i$ and a "right" copy $Q'_i$). We wire the two sets of subqueries $Q_1, \ldots, Q_m, Q'_1, \ldots, Q'_m$ as follows.

Consider first the integer function $NAND(k, l)$ defined by Table II. We can enforce that two variables, $x$ and $y$, labeled $L_k$ and $L_l$ in their respective subqueries, cannot both match the topmost node labeled $L_k$ resp. $L_l$ in the left, respective right,

| $k\backslash l$ | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 10 | 13 | 18 |
| 2 | 5 | 8 | 13 |
| 3 | 2 | 5 | 10 |

Table II. The function $NAND(k,l)$.

part of the data tree by adding an atom of the form $\mathit{Following}^{NAND(k,l)}(x,y)$ to the query.

For each pair of clauses $C_i$, $C_j$, variable $x$ such that $Q_i$ (resp., $Q'_i$) contains the unary atom $L_k(x)$, and variable $y$ such that $Q'_j$ (resp., $Q_j$) contains the unary atom $L_l(y)$, if

—the $k$-th literal of $C_i$ occurs also in $C_j$ and

—the $k$-th literal of $C_i$ and the $l$-th literal of $C_j$ are different,

then we add an atom $\mathit{Following}^{NAND(k,l)}(x,y)$ to the query.

These query atoms make sure that if a literal is chosen to be true in one clause, it must be selected to be true in all other clauses as well. In the case that $i = j$, the idea is to make sure that both copies of the query gadget of each clause, $Q_i$ and $Q'_i$, make the same choice of selected literal. The case that $i \neq j$ models the interaction between distinct clauses. Thus our query assures that each literal is assigned the same truth value in all clauses.

Using two copies of the query gadget for each clause and two copies of the tree gadget of Figure 5 (a) in the data tree is necessary, as we cannot use $\mathit{Following}^k$-atoms to make sure that two variables are not both assigned their topmost positions in the data tree (corresponding to "true") if the data tree consists just of the tree of Figure 5 (a) and these two topmost positions in the data tree coincide.

This concludes the construction, which can be easily implemented to run in logarithmic space. It is not difficult to verify that the fixed data tree satisfies the query precisely if the 1-in-3 3SAT instance is satisfiable. □

THEOREM 5.3. *Conjunctive queries over the signatures*

$$\tau_7 := \langle (\mathit{Label}_a)_{a\in\Sigma}, \mathit{Child}^+, \mathit{Following}\rangle,$$
$$\tau_8 := \langle (\mathit{Label}_a)_{a\in\Sigma}, \mathit{Child}^*, \mathit{Following}\rangle$$

*are NP-complete w.r.t. query complexity.*

PROOF. The same encoding as in the previous proof can be used, with the only difference that $\mathit{Child}^*$ resp. $\mathit{Child}^+$ is used instead of *Child* in the query. In fact, if the topmost position for $v_1$ (resp. $v_2$, $v_3$) is chosen, there are two possible matches for "A" (resp. three for "B" and two for "C"). This has no impact on the constraints across clauses or the constraints that at most one variable of each clause is assigned to its topmost position. To make sure that *at least* one variable of each clause is assigned its topmost position, the constraints of the query assure that either "A", "B", or "C" are assigned to the correspondingly labeled node at depth two in the subtree of the clause (rather than depth three). □

Since *Following* can be defined by a conjunctive query over $\mathit{Child}^*$ and $\mathit{NextSibling}^+$ (see Equation (1) in Section 2),

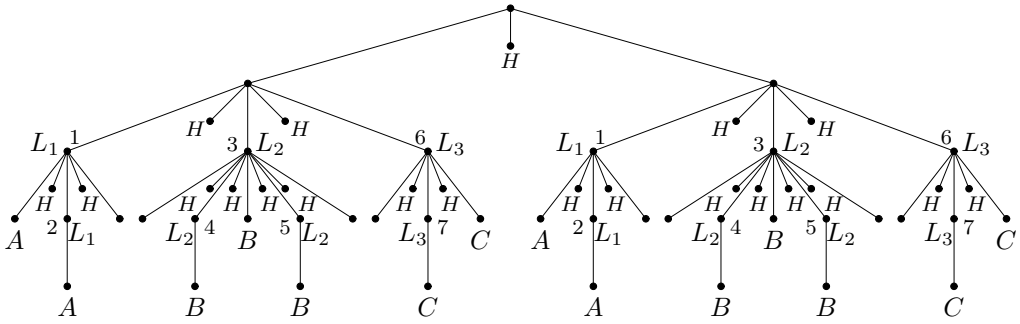



Fig. 6. Data tree of proof of Theorem 5.6.

COROLLARY 5.4. *Conjunctive queries over the signature*

$$\tau_9 := \langle (Label_a)_{a\in\Sigma}, Child^*, NextSibling^+ \rangle$$

*are NP-complete w.r.t. query complexity.*

THEOREM 5.5. *Conjunctive queries over the signature*

$$\tau_{10} := \langle (Label_a)_{a\in\Sigma}, Child^*, NextSibling \rangle$$

*are NP-complete w.r.t. query complexity.*

PROOF. If we replace *Following* by

$$Following'(x,y) := \exists z_1 \exists z_2 \; Child^*(z_1,x) \wedge NextSibling(z_1,z_2) \wedge Child^*(z_2,y),$$

we can reuse the construction of the proof of Theorem 5.2 (in the modified form of the proof of Theorem 5.3). □

THEOREM 5.6. *Conjunctive queries over the signature*

$$\tau_{11} := \langle (Label_a)_{a\in\Sigma}, Child^*, NextSibling^* \rangle$$

*are NP-complete w.r.t. query complexity.*

PROOF. The proof basically uses the same argument as Corollary 5.4. However, to deal with $NextSibling^*$ rather than $NextSibling^+$, we need a way to ensure that $NextSibling^*$ moves at least one step to the right. We thus replace each occurrence of *Following* in the construction of the proof of Theorem 5.2 by

$$Following'(x,y) := \exists z_1 \exists z_2 \exists z_3 \; Child^*(z_1,x) \wedge NextSibling^*(z_1,z_2) \wedge H(z_2) \wedge NextSibling^*(z_2,z_3) \wedge Child^*(z_3,y).$$

The modified data tree is as shown in Figure 6. It uses specially labeled auxiliary nodes inserted between each pair of adjacent siblings in the data tree of the proof of Theorem 5.2. □

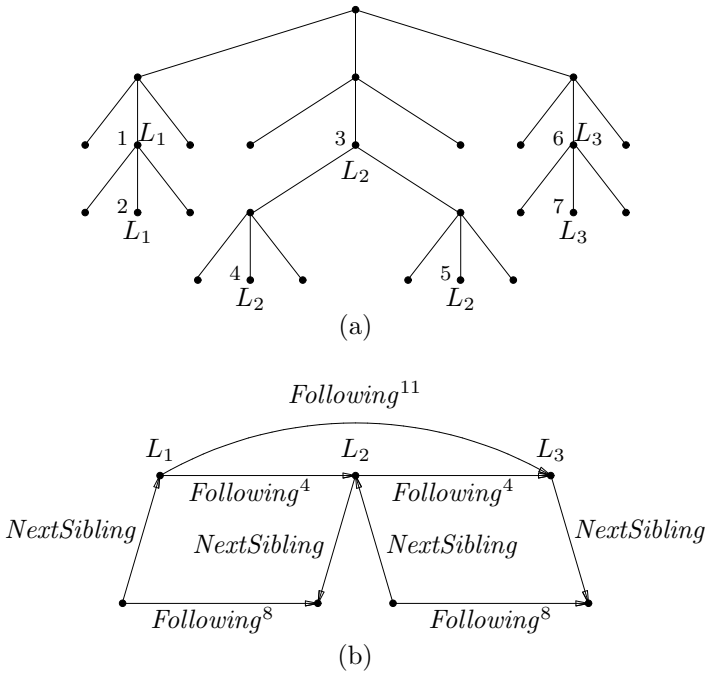



Fig. 7. Encoding the selection of exactly one of the positive literals of a clause as a conjunctive query over signature $\tau_{15}$.

THEOREM 5.7. *Conjunctive queries over the signatures*

$$\begin{aligned}\tau_{12} &:= \langle (Label_a)_{a\in\Sigma}, Child^+, NextSibling\rangle,\\ \tau_{13} &:= \langle (Label_a)_{a\in\Sigma}, Child^+, NextSibling^+\rangle,\\ \tau_{14} &:= \langle (Label_a)_{a\in\Sigma}, Child^+, NextSibling^*\rangle\end{aligned}$$

*are NP-complete w.r.t. query complexity.*

PROOF. The proofs are analogous to the proofs for the respective signatures with $Child^*$ rather than $Child^+$, except that we modify the respective data trees as follows: Each edge $\langle u, w\rangle$ is replaced by two edges $\langle u, v\rangle, \langle v, w\rangle$, where $v$ is a new node. Now, to make a *Following*-step between two nodes corresponding to original tree nodes, we can use the relation

$$Following''(x, y) := \exists z_1 \exists z_2 \exists z_3\ Child^+(z_1, x) \wedge NextSibling^\alpha(z_2, z_3) \wedge Child^+(z_3, y).$$

where $\alpha$ is "1" for $\tau_{12}$, "+" for $\tau_{13}$, and "$*$" for $\tau_{14}$. □

THEOREM 5.8. *Conjunctive queries over the signatures*

$$\begin{aligned}\tau_{15} &:= \langle (Label_a)_{a\in\Sigma}, Following, NextSibling\rangle,\\ \tau_{16} &:= \langle (Label_a)_{a\in\Sigma}, Following, NextSibling^+\rangle,\\ \tau_{17} &:= \langle (Label_a)_{a\in\Sigma}, Following, NextSibling^*\rangle\end{aligned}$$

*are NP-complete w.r.t. query complexity.*

PROOF. We first look at signature $\tau_{15}$. Consider the data tree shown in Figure 7 (a) and the query of Figure 7 (b).

As in the proof of Theorem 5.2, there is again one variable per label $L_1$ ($L_2$, $L_3$), which we call $v_1$ ($v_2$, $v_3$). Again, at most one variable $v_1$, $v_2$, and $v_3$ can be mapped to its topmost position. The query shown in Figure 7 (a) requires that precisely the partial assignments

$$\theta(v_1) := 1,\ \theta(v_2) := 4,\ \theta(v_3) := 7$$
$$\theta(v_1) := 2,\ \theta(v_2) := 3,\ \theta(v_3) := 7$$
$$\theta(v_1) := 2,\ \theta(v_2) := 5,\ \theta(v_3) := 6$$

can be extended to solutions of the query.

This provides us with an encoding for the selection of exactly one literal from a given clause with three positive literals. The full reduction from 1-in-3 3SAT over positive literals can be obtained analogously to the proof of Theorem 5.2.

The same reduction can be used to prove the corresponding result for the signatures $\tau_{16}$ and $\tau_{17}$. □

## 6. EXPRESSIVENESS

In this section, we study the expressive power of conjunctive queries over trees. The main result is that for each conjunctive query over trees, an equivalent *acyclic positive query* (APQ) can be found. However, these APQs are in general exponentially larger. As we show in Section 7, this is necessarily so.

We introduce a number of technical notions. In Section 2, query graphs were introduced as directed (multi)graphs. Below, we will deal with two kinds of cycles in query graphs; *directed cycles*, the standard notion of cycles in directed graphs, and the more general *undirected cycles*, which are cycles in the undirected *shadows* of query graphs.[8] The standard notion of conjunctive query acyclicity in the case that relations are at most binary refers to the absence of undirected cycles from the shadow of the query graph.

Let $F \subseteq \mathbf{Ax}$ be a set of axes. We denote by CQ[$F$] the conjunctive queries over signature $\langle (Label_a)_{a\in\Sigma}, F\rangle$. By PQ[$F$] we denote the positive (first-order) queries (written as finite unions of conjunctive queries) over $F$. We denote the acyclic positive queries – that is, unions of *acyclic* conjunctive queries – over $F$ by APQ[$F$].

REMARK 6.1. Given a set of XPath axes $F$, let $F^{-1}$ denote their inverses (e.g., *Parent* for *Child*; see [World Wide Web Consortium 1999] for the names of the inverse XPath axes). It is easy to show that for any set $F$ of XPath axes, positive Core XPath[$F \cup F^{-1}$], the positive, navigation-only fragment of the XPath language [Gottlob et al. 2005], captures the unary APQ[$F$] on trees in which each node has (at most) one label. No proof of this is presented here because a formal definition of XPath is tedious and the result follows immediately from such a definition. (Positive Core XPath queries are acyclic and support logical disjunction.)

Before we can get to the main result of this section, Theorem 6.6, we need to define the notion of a *join lifter*, for which we will subsequently give an intuition

[8]The shadow of a directed graph is obtained by replacing each directed edge from node $u$ to node $v$ by an undirected edge between $u$ and $v$.

and an example. After providing two lemmata, we will be able to prove Theorem 6.6. The proof of the main result employs a rewrite system whose workings are illustrated in a detailed example in Figure 8 (Example 6.8). The reader may find it helpful to start with that example before reading on sequentially from here.

DEFINITION 6.2. Let $F$ be a set of binary relations. A positive quantifier-free formula $\psi_{R,S}(x, y, z)$ in Disjunctive Normal Form (DNF) is called a *join lifter* over $F$ for binary relations $R$ and $S$ if

(1) each conjunction of $\psi_{R,S}(x, y, z)$ is of one of the following five forms:
   (a) $P(x, y) \wedge P'(y, z)$
   (b) $P(y, x) \wedge P'(x, z)$
   (c) $P(x, z) \wedge y = z$
   (d) $P(y, z) \wedge x = z$
   (e) $P(x, z) \wedge x = y$
   where $P, P' \in F$ and

(2) for all trees $\mathcal{A}$ and nodes $a, b, c$,

$$(\mathcal{A}, a, b, c) \vDash \phi_{R,S}[a, b, c] \;\Leftrightarrow\; (\mathcal{A}, a, b, c) \vDash \psi_{R,S}[a, b, c].$$

   where $\phi_{R,S}(x, y, z) = R(x, z) \wedge S(y, z)$.
   (Subsequently, we will write this as $\psi_{R,S} \equiv \phi_{R,S}$.)

A join lifter $\psi_{R,S}$ can be used to rewrite a conjunctive query $Q$ that contains atoms $R(x, z), S(y, z)$ – the role of such pairs of atoms will be clarified below, in the proof of Lemma 6.5– into a union of conjunctive queries (one conjunctive query for each conjunction $C$ of the DNF formula $\psi_{R,S}$, by replacing $R(x, z), S(y, z)$ by $C$) such that none of the conjunctive queries obtained is larger than $Q$. In fact, each of conjunctive queries obtained is either shortened (because equality atoms $v = w$ in conjunctions of form (c), (d) or (e) can be eliminated after substituting variable $v$ by $w$ everywhere in the query) or the join on $z$ is intuitively lifted "up" in the query graph using a conjunction of form (a) or (b).

EXAMPLE 6.3. The formula

$$\psi_{\mathit{Child},\mathit{NextSibling}}(x, y, z) = \mathit{Child}(x, y) \wedge \mathit{NextSibling}(y, z)$$

is a join lifter for *Child* and *NextSibling* because it satisfies the syntactic requirement (1) – the formula is a single conjunction of form (a) – and the equivalence (2)

$$\psi_{\mathit{Child},\mathit{NextSibling}}(x, y, z) \equiv \phi_{\mathit{Child},\mathit{NextSibling}}(x, y, z) = \mathit{Child}(x, z) \wedge \mathit{NextSibling}(y, z).$$

of Definition 6.2. Conjunctions of form (a) such as this one lift the join occurring in $\phi_{\mathit{Child},\mathit{NextSibling}}$ one level up in the query graph – here from variable $z$ in $\phi_{\mathit{Child},\mathit{NextSibling}}$ to variable $y$ in $\psi_{\mathit{Child},\mathit{NextSibling}}$ when rewriting $\phi_{\mathit{Child},\mathit{NextSibling}}$ by $\psi_{\mathit{Child},\mathit{NextSibling}}$.

Moving joins *upward* is only meaningful in queries whose query graphs do not have directed cycles. As demonstrated by the following lemma, such cycles can always be eliminated.

LEMMA 6.4. *Let $Q$ be a $CQ[\mathbf{Ax}]$ that contains a directed cycle*

$$R_1(x_1, x_2), R_2(x_2, x_3), \ldots, R_{k-1}(x_{k-1}, x_k), R_k(x_k, x_1).$$

*If $R_1, \ldots, R_k \in \{\mathit{Child}^*, \mathit{NextSibling}^*\}$, then $Q$ is equivalent to the query obtained by adding $x_1 = x_2 = \cdots = x_k$ to the body of $Q$. Otherwise, $Q$ is unsatisfiable.*

PROOF. The graph of the relation $\mathit{Child} \cup \mathit{NextSibling} \cup \mathit{Following}$ is acyclic. Therefore, a query with a cycle can only be satisfied if all variables in the cycle are mapped to the same node. If the cycle contains an irreflexive axis (any axis besides $\mathit{Child}^*$ and $\mathit{NextSibling}^*$), the query is unsatisfiable. □

LEMMA 6.5. *Let $F \subseteq \mathbf{Ax}$ be a set of axes and let there be join lifters $\psi_{R,S}$ over $F$ for each pair $(R, S)$ of relations in $F$. Then, each $CQ[F]$ can be rewritten into an equivalent $APQ[F]$ in singly exponential time.*

PROOF. Given a conjunctive query $Q_0$, we execute the following algorithm. Let $\mathcal{Q}$ be a set of conjunctive queries, initially $\{Q_0\}$. Repeat the following until the query graphs of all queries in $\mathcal{Q}$ are forests.

(1) Choose any conjunctive query $Q$ from $\mathcal{Q}$ whose query graph is not a forest.

(2) If $Q$ contains a directed cycle in which a predicate other than $\mathit{NextSibling}^*$ or $\mathit{Child}^*$ appears, $Q$ is unsatisfiable (by Lemma 6.4) and is removed from $\mathcal{Q}$.

(3) For each directed cycle in $Q$ that consists exclusively of $\mathit{Child}^*$ and $\mathit{NextSibling}^*$ atoms, we identify the variables occurring in it. (That is, if $x_1, \ldots, x_n$ are precisely all the variables of the cycle, we replace each occurrence of any of these variables in the body *or head* of $Q$ by $x_1$.) Atoms of the form $\mathit{Child}^*(x_1, x_1)$ or $\mathit{NextSibling}^*(x_1, x_1)$ are removed.
In order to assure safety, we add an atom $\mathit{Node}(x_1)$ if $x_1$ now does not occur in any remaining atom. (The predicate *Node* matches any node and can be defined as $R(x_1, x_1')$, where $R$ is a predicate of the directed cycle just eliminated – either $\mathit{Child}^*$ or $\mathit{NextSibling}^*$ – and $x_1'$ is a new variable.)
By Lemma 6.4, the outcome of this transformation is equivalent to the input query.

(4) Now there are no directed cycles left in the query graph, but undirected cycles may remain. If $Q$ contains undirected cycles, we choose a variable $z$ that is in an undirected cycle such that there is no directed path in the query graph leading from $z$ to another variable that is in an undirected cycle as well. (Such a choice is possible because there are no directed cycles in the query graph.) The cycle contains two atoms $R(x, z), S(y, z)$.
Now, we use join lifter $\psi_{R,S}$ to replace these two atoms. Let $\psi_{R,S}$ be the DNF $\psi_{R,S}^{(1)} \vee \cdots \vee \psi_{R,S}^{(k)}$ such that the $\psi_{R,S}^{(i)}$ are conjunctions of atoms. We create copies $Q_1, \ldots, Q_k$ of $Q$ and replace $R(x, z), S(y, z)$ in each $Q_i$ by $\psi_{R,S}^{(i)}$. If $\psi_{R,S}^{(i)}$ contains an equality atom $v = w$, we replace each occurrence of variable $w$ in $Q_i$ by $v$ and remove the equality atom. Finally, we replace $Q$ in $\mathcal{Q}$ by $Q_1, \ldots, Q_k$.

First we show that this algorithm indeed terminates. The elimination of directed cycles – steps (2) and (3) – is straightforward, but we need to consider in more detail how the algorithm deals with undirected cycles. The idea here is to eliminate

undirected cycles from the bottom to the top (with respect to the direction of edges in the query graph.) This is done by rewriting bottom atoms $R(x,z), S(y,z)$ of undirected cycles using the join lifters $\psi_{R,S}$. While $R(x,z), S(y,z)$ are two binary atoms that involve $z$, each conjunction in join lifter $\psi_{R,S}$ contains only one binary atom over $z$ apart from a possible equality atom. Therefore, each rewrite step either removes $z$ from at least one cycle or identifies $z$ with either $x$ or $y$ via an equality atom (which, for our purposes, means to remove $z$ entirely, and thus also from any cycle it appears in).

Let $|V|$ be the number of variables and $|E|$ be the number of binary atoms in $Q_0$. The number of atoms in a conjunctive query never increases by the rewrite steps (each conjunction of the formulae $\psi_{R,S}$ is of length two). For a given bottommost variable $z$ of the query graph that is in an undirected cycle, there can be at most $|E|$ incoming edges (i.e., binary atoms) for $z$. After at most $|E|-1$ appropriate iterations of our algorithm, there is only one incoming edge for $z$ or $z$ has been eliminated. Consequently, after no more than $|V| \cdot |E|$ iterations of our algorithm on a conjunctive query (in each of which a join lifter can be applied), the conjunctive query is necessarily acyclic.

In each such loop, a single query may be replaced by at most $k$ others, where $k$ is the maximum number of conjunctions occurring in a join lifter – a constant (no greater than three in this article). Thus, we make no more than $k^{|V|\cdot|E|}$ iterations in total until all conjunctive queries in $\mathcal{Q}$ are acyclic, i.e. their query graphs are forests. This is the termination condition of our algorithm.

Thus, $\mathcal{Q}$ cannot contain more than $k^{|V|\cdot|E|}$ conjunctive queries, all of size $\leq |Q_0|$. Since the cycle detection and transformation procedures in (2) to (4) can be easily implemented to run in polynomial time each, the overall running time of our algorithm is singly exponential.

The query computed by the algorithm is equivalent to $Q_0$. This follows by induction from the fact that the steps (2) to (4) each produce equivalent rewritings. (The individual arguments are provided with steps (2) to (4).) Thus, on termination, $\mathcal{Q}$ is a union of acyclic conjunctive queries – an APQ – equivalent to $Q_0$.

Note that step (4) can introduce new directed cycles into a query; therefore, it may be necessary to repeat steps (2) and (3) after an application of step (4), as done by our algorithm. □

Note that the rewriting technique of the previous algorithm is *nondeterministic* (by the choice of next query to rewrite in step (1)), but we do not prove confluence of our rewrite system since it is not essential to our main theorem, stated next.

THEOREM 6.6. *(1) For* $F \subseteq \{\mathit{Child}, \mathit{Child}^*, \mathit{Child}^+\}$, $CQ[F] \subseteq APQ[F]$.
*(2) For* $F \subseteq \{\mathit{Child}, \mathit{Child}^+, \mathit{NextSibling}, \mathit{NextSibling}^*, \mathit{NextSibling}^+\}$,

$$CQ[F] \subseteq APQ[F].$$

*(3) For* $F \subseteq \{\mathit{Child}, \mathit{Child}^*, \mathit{Child}^+, \mathit{NextSibling}, \mathit{NextSibling}^*, \mathit{NextSibling}^+\}$,

$$CQ[F] \subseteq APQ[F \cup \{\mathit{Child}^+\}].$$

PROOF. Consider the DNF formulae

$$\psi_{R,S}(x,y,z) = \begin{cases} R(x,z) \wedge x = y & \ldots\ R = S \in \{\mathit{Child}, \mathit{NextSibling}\}, \\ (R(x,z) \wedge R(y,x)) \vee (R(x,y) \wedge R(y,z)) & \ldots\ R = S \in \{\mathit{Child}^*, \mathit{NextSibling}^*\} \\ (R(x,z) \wedge R(y,x)) \vee (R(x,y) \wedge R(y,z)) \vee (R(x,z) \wedge x = y) & \ldots\ R = S \in \{\mathit{Child}^+, \mathit{NextSibling}^+\} \\ (R(x,z) \wedge y = z) \vee (R(x,z) \wedge S(y,x)) & \ldots\ R \in \{\mathit{Child}, \mathit{NextSibling}\},\ S = R^* \\ (R(x,z) \wedge x = y) \vee (R(x,z) \wedge S(y,x)) & \ldots\ R \in \{\mathit{Child}, \mathit{NextSibling}\},\ S = R^+ \\ (R(x,z) \wedge y = z) \vee (R(x,z) \wedge S(y,x)) \vee (R(y,z) \wedge S(x,y)) & \ldots\ R = \chi^+,\ S = \chi^* \text{ where } \chi \in \{\mathit{Child}, \mathit{NextSibling}\} \\ R(x,z) \wedge S(y,x) & \ldots\ R \in \{N, N^*, N^+\}, N = \mathit{NextSibling}, S \in \{\mathit{Child}, \mathit{Child}^+\} \\ (R(x,z) \wedge y = z) \vee (R(x,z) \wedge \mathit{Child}^+(y,x)) & \ldots\ R \in \{N, N^*, N^+\}, N = \mathit{NextSibling}, S = \mathit{Child}^* \\ \psi_{S,R}(y,x,z) & \ldots\ \text{otherwise} \end{cases}$$

which are defined for all

$$R, S \in \{\mathit{Child}, \mathit{Child}^*, \mathit{Child}^+, \mathit{NextSibling}, \mathit{NextSibling}^*, \mathit{NextSibling}^+\}.$$

The $\psi_{R,S}$ are join lifters for each $R, S$. The syntactic properties of join lifters of Definition 6.2 can be easily verified by inspection. Moreover, indeed for all $R$, $S$, $\psi_{R,S}(x,y,z) \equiv \phi_{R,S}(x,y,z) = R(x,z) \wedge S(y,z)$. The arguments required to show this are very simple and are omitted. (For example, $\phi_{\mathit{Child},\mathit{Child}} \equiv \mathit{Child}(x,z) \wedge x = y$ because each node in a tree can have only at most one parent.)

Thus, the $\psi_{R,S}$ are indeed join lifters. Now observe that $\psi_{R,\mathit{Child}^*}$ for $R \in \{\mathit{NextSibling}, \mathit{NextSibling}^+, \mathit{NextSibling}^*\}$ uses the $\mathit{Child}^+$ axis, but all other $\psi_{R,S}$ only use the relations $R$ and $S$ (plus equality). From Lemma 6.5, it follows that for $F$ such that $\mathit{Child}^* \notin F$ or $\mathit{NextSibling}, \mathit{NextSibling}^+, \mathit{NextSibling}^* \notin F$, each $CQ[F]$ can be translated into an equivalent $APQ[F]$ (parts 1 and 2 of our theorem) and otherwise, each $CQ[F]$ can be translated into an equivalent $APQ[F \cup \{\mathit{Child}^+\}]$ (part 3). □

In all three cases of Theorem 6.6, the conjunctive queries can be rewritten into equivalent APQs in singly exponential time.

Similar techniques to those of the previous two proofs were used in [Olteanu et al. 2002] to eliminate backward axes from XPath expressions and in [Schwentick 2000] to rewrite first-order queries over trees given by certain regular path relations. The special cases of Theorem 6.6 that $CQ[\{\mathit{Child}\}] \subseteq APQ[\{\mathit{Child}\}]$ and $CQ[\{\mathit{Child}, \mathit{Child}^*\}] \subseteq APQ[\{\mathit{Child}, \mathit{Child}^*\}]$ are implicit in [Benedikt et al. 2003].

EXAMPLE 6.7. Consider the query $Q_0(x,y) \leftarrow \mathit{Child}^*(x,y) \wedge \mathit{NextSibling}^*(x,y)$.

Since $\psi_{NextSibling^*,Child^*}(x,x,y) = (NextSibling^*(x,y) \wedge x = y) \vee (NextSibling^*(x,y) \wedge Child^+(x,x))$, we set $\mathcal{Q} = \{Q, Q'\}$ with $Q(x,x) \leftarrow NextSibling^*(x,x)$, which is further simplified to $Q(x,x) \leftarrow Node(x)$, and $Q'(x,y) \leftarrow NextSibling^*(x,y) \wedge Child^+(x,x))$. $Q'$ is unsatisfiable due to the directed cycle defined by its second atom and is removed from $\mathcal{Q}$. We obtain the APQ $\{Q\}$ which is equivalent to $Q_0$.

EXAMPLE 6.8. Figure 8 illustrates the query rewriting algorithm of the proof of Lemma 6.5 using the join lifters of the proof of Theorem 6.6 by means of an example. The example query $Q$ is that from the introduction, but since Theorem 6.6 does not handle the *Following* axis, we first rewrite it using $Child^*$ and $NextSibling^+$. All conjunctive queries that we obtain are unsatisfiable, except for one, shown at the bottom left corner of Figure 8. Thus, for $Q$ there exists an equivalent acyclic *conjunctive* query.

Note that in Figure 8 we make an exception from the conventions followed throughout this article by labeling the nodes of the query graphs with the variable names in order to allow for the variables to be tracked through the rewrite steps more easily. □

We complement Theorem 6.6 by two further translation theorems.

THEOREM 6.9. *If $Q$ is a $CQ[F]$ such that*

$$F \subseteq \{Child, NextSibling, NextSibling^*, NextSibling^+, Following\},$$

*then $Q$ can be rewritten into an equivalent $APQ[F \cup \{NextSibling^+\}]$ in singly exponential time.*

PROOF. We extend $\psi_{R,S}$ from the proof of Theorem 6.6 by join lifter formulae for $S = Following$ and $R \in F$:

$$\begin{aligned}
\psi_{NextSibling,Following}(x,y,z) &:= (NextSibling(x,z) \wedge x = y) \vee \\
&\quad (NextSibling(x,z) \wedge Following(y,x)) \\
\psi_{NextSibling^+,Following}(x,y,z) &:= (NextSibling^+(x,z) \wedge x = y) \vee \\
&\quad (NextSibling^+(x,z) \wedge Following(y,x)) \vee \\
&\quad (NextSibling^+(x,y) \wedge NextSibling^+(y,z)) \\
\psi_{NextSibling^*,Following}(x,y,z) &:= (NextSibling^*(x,z) \wedge Following(y,x)) \vee \\
&\quad (NextSibling^*(x,y) \wedge NextSibling^+(y,z)) \\
\psi_{Child,Following}(x,y,z) &:= (Child(x,z) \wedge x = y) \vee \\
&\quad (Child(x,z) \wedge Following(y,x)) \vee \\
&\quad (Child(x,y) \wedge NextSibling^+(y,z)) \\
\\
\psi_{Following,Following}(x,y,z) &:= (Following(x,z) \wedge x = y) \vee \\
&\quad (Following(x,z) \wedge Following(y,x)) \vee \\
&\quad (Following(x,y) \wedge Following(y,z))
\end{aligned}$$

Each $\psi_{R,S}$ is defined using only relations $R, S, =$ and $NextSibling^+$. Now the theorem follows immediately from Lemma 6.5. □

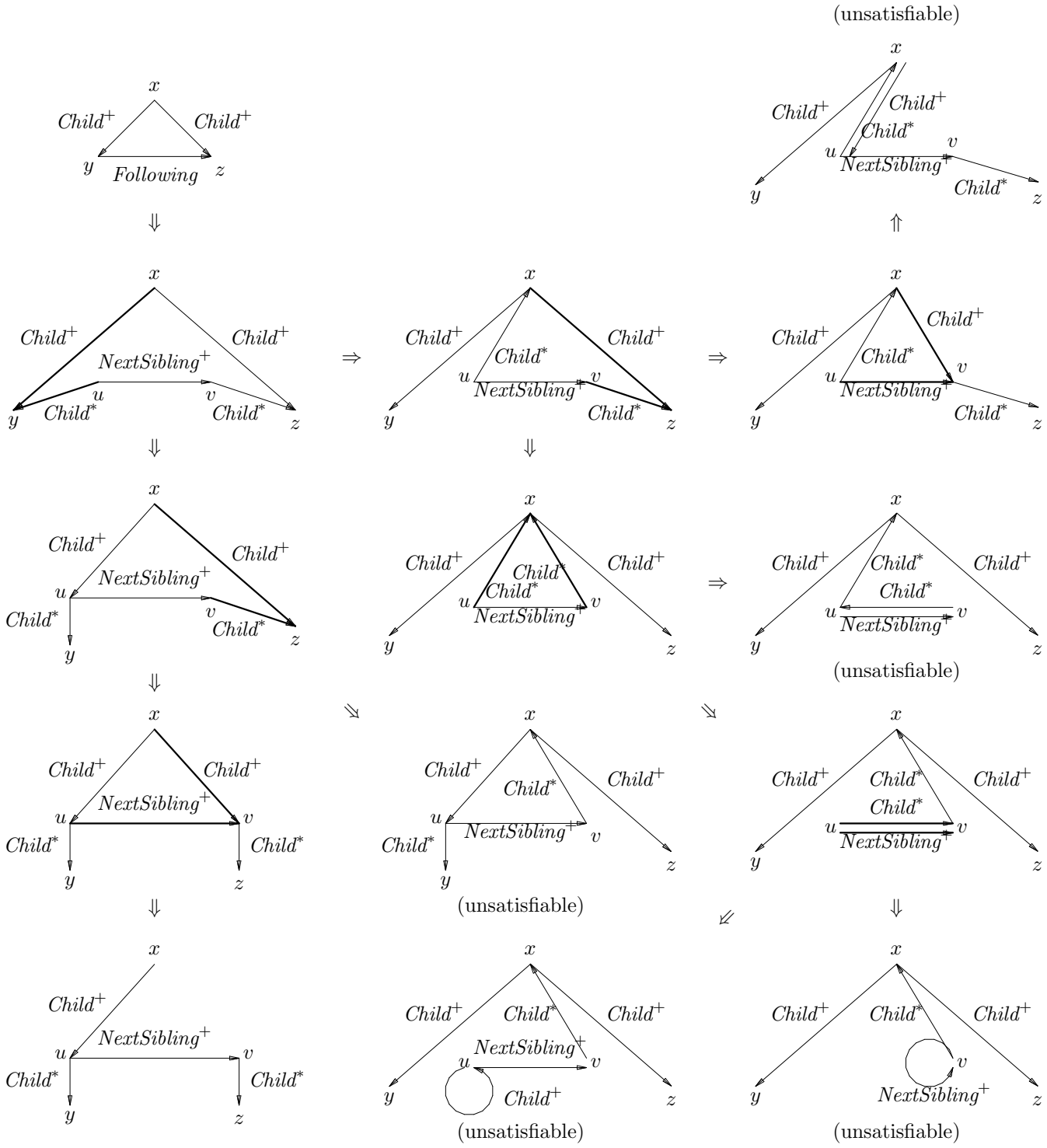



Fig. 8. Translation of a conjunctive query into an APQ.

THEOREM 6.10. *If $Q$ is a $CQ[F]$ such that $F \subseteq \mathbf{Ax}$, then $Q$ can be rewritten into an equivalent $APQ[F \cup \{\mathit{Child}^+, \mathit{NextSibling}^+\}]$ in singly exponential time.*

PROOF. Given query $Q$, we first rewrite all occurrences of *Following* using $\mathit{Child}^*$ and $\mathit{NextSibling}^+$ using Equation (1) from Section 2. In order to be economical with axes, we rewrite all $n$ occurrences of $\mathit{Child}^*$ using $\mathit{Child}^+$. We define an APQ consisting of $2^n$ copies of $Q$ such that in the $m$-th copy of $Q$, the $k$-th $\mathit{Child}^*(x, y)$ atom is replaced by $\mathit{Child}^+(x, y)$ if the $k$-th bit of $m$ represented in binary is, say, 1 and by $x = y$ otherwise (that is, all occurrences of variable $y$ in the query are replaced by $x$). Clearly, since $\mathit{Child}^*(x, y) \Leftrightarrow \mathit{Child}^+(x, y) \vee x = y$, the APQ obtained in this way is equivalent to $Q$. Then we apply the algorithm of the proof of Lemma 6.5 using the join lifters as in the proof of Theorem 6.6 (3) to each of the $2^n$ modified conjunctive queries and compute the union of the APQs obtained. Of course, the overall transformation can be again effected in exponential time. □

It follows that the acyclic positive queries capture the positive queries over trees.

COROLLARY 6.11. $PQ[\mathbf{Ax}] = APQ[\mathbf{Ax}]$.

REMARK 6.12. Since $\mathit{Child}^+$ and $\mathit{NextSibling}^+$ are XPath axes ("descendant" and "following-sibling"), it follows from Theorem 6.10 that each unary conjunctive query over XPath axes can also be formulated as an XPath query. This is in contrast to full first-order logic (i.e., with negation) on trees, which is known to be stronger than acyclic first-order logic on trees resp. Core XPath [Marx 2005].

Obviously, the $CQ[F]$ are not closed under union. On trees of one node only, conjunctive queries are equivalent to ones which do not use binary atoms. It is easy to see that the query $\{x \mid A(x) \vee B(x)\}$ has no conjunctive counterpart.

PROPOSITION 6.13. *For any $F \subseteq \mathbf{Ax}$, $CQ[F] \neq APQ[F]$.*

There are signatures with axes for which all conjunctive queries can be rewritten into APQ's in polynomial time.[9]

PROPOSITION 6.14 [GOTTLOB AND KOCH 2004]. *Any $CQ[\{\mathit{Child}, \mathit{NextSibling}\}]$ can be rewritten into an equivalent acyclic $CQ[\{\mathit{Child}, \mathit{NextSibling}, \mathit{NextSibling}^*\}]$ in linear time.*

REMARK 6.15. It is easy to verify by inspecting the proof in [Gottlob and Koch 2004] that rewriting each CQ[*Child*, *NextSibling*] into an equivalent acyclic CQ[*Child*, *NextSibling*] in linear time is also possible. (The proof there also deals with relations such as *FirstChild*. If these are not present, $\mathit{NextSibling}^*$ is not required.)

## 7. SUCCINCTNESS

The translations from conjunctive queries into APQs of the Theorems 6.6, 6.9 and 6.10 run in exponential time and can produce APQs of exponential size. In this section, we show that this situation cannot be improved upon: there are conjunctive queries over trees that cannot be polynomially translated into equivalent APQs.

[9] As shown in the next section, there are also signatures for which this is not possible.

By the size $|Q|$ of a Boolean conjunctive query $Q$, we denote the number of atoms in its body. The size of an APQ is given by the sum of the sizes of the constituent conjunctive queries.

Let $D_n$ denote the $n$-diamond Boolean conjunctive query

$$D_n \leftarrow Y_1(y_1) \wedge \bigwedge_{i=1}^{n} \big( \mathit{Child}^+(y_i, x_i) \wedge X_i(x_i) \wedge \mathit{Child}^+(x_i, y_{i+1}) \wedge \mathit{Child}^+(y_i, x'_i) \wedge X'_i(x'_i) \wedge \mathit{Child}^+(x'_i, y_{i+1}) \wedge Y_{i+1}(y_{i+1}) \big).$$

A graphical representation of $D_n$ is provided in Figure 9 (a).

The following is the main result of this section:

THEOREM 7.1. *There is no family $(\mathcal{Q}_n)_{n\geq 1}$ of queries in $APQ[\mathbf{Ax}]$ such that each $\mathcal{Q}_n$ is of size polynomial in $n$ and is equivalent to $D_n$.*

Before we can show this, we have to provide a few definitions.

We use the acronyms ABCQ for *acyclic Boolean conjunctive queries* and DABCQ (*directed ABCQ*) for Boolean conjunctive queries whose query graphs are acyclic. That is, the query graph of a DABCQ is a directed acyclic graph, while the query graph of an ABCQ is a forest (because conjunctive query acyclicity is defined with respect to the undirected shadows of query graphs). By Lemma 6.4, an equivalent $DABCQ[F]$ exists (and can be computed efficiently) for each Boolean $CQ[F]$ that is satisfiable, for any $F \subseteq \mathbf{Ax}$. The queries $D_n$ are $DABCQ[\{\mathit{Child}^+\}]$.

For a DABCQ $Q$, let $\Pi_Q \subseteq \mathit{Var}(Q)^*$ denote the set of *variable-paths* in the query graph of $Q$ from variables that have in-degree zero to variables that have out-degree zero. For example, if $Q$ is the left- and bottommost query of Figure 8, then $\Pi(Q) = \{xuy, xuvz\}$. We say that a label $L$ *occurs* in variable-path $\pi \in \Pi_Q$ iff there is a variable $x$ in $\pi$ for which $Q$ contains a unary atom $L(x)$.

By a *path-structure*, we denote a tree structure in which the graph of the *Child*-relation is a path. Given a variable-path $x_1.\cdots.x_k$, the associated *label-path* is the path-structure of $k$ nodes in which the $i$-th node is labeled $L$ iff $Q$ contains atom $L(x_i)$. Observe that some nodes of this structure may be unlabeled, and some may have several labels. Given a set $P$ of variable-paths, let $LP(P)$ denote the corresponding label-paths.

We say that a path-structure is *$k$-scattered* if (*) it consists of at least $k$ nodes, (*) each node has at most one label, (*) no two nodes have the same label, and (*) if node $v$ has a label and node $v'$ ($v \neq v'$) either is the topmost node, the bottommost node, or has a label, then the distance between $v$ and $v'$ is at least $k$.

In order to prove our theorem, we need two technical lemmata. The first states, *essentially*, that on sufficiently scattered path structures, each ABCQ is equivalent to an ABCQ that only uses the axes $\mathit{Child}^+$ and $\mathit{Child}^*$. This is somewhat reminiscent of results on the locality of first-order queries (cf. e.g. [Libkin 2004]).

LEMMA 7.2. *Let $Q$ be an $ABCQ[\mathbf{Ax}]$ that is true on at least one $|Q|$-scattered path-structure. Then, there is an $ABCQ[\{\mathit{Child}^+, \mathit{Child}^*\}]$ $Q'$ such that $Q' \subseteq Q$, $|Q'| \leq |Q|$, and $Q'$ is true on all $|Q|$-scattered path-structures on which $Q$ is true.*

The second lemma states that two $DABCQ[\{\mathit{Child}^*, \mathit{Child}^+\}]$ $Q$ and $Q'$ with $LP(\Pi_Q) \neq LP(\Pi_{Q'})$ differ in the sets of path structures on which they are true.

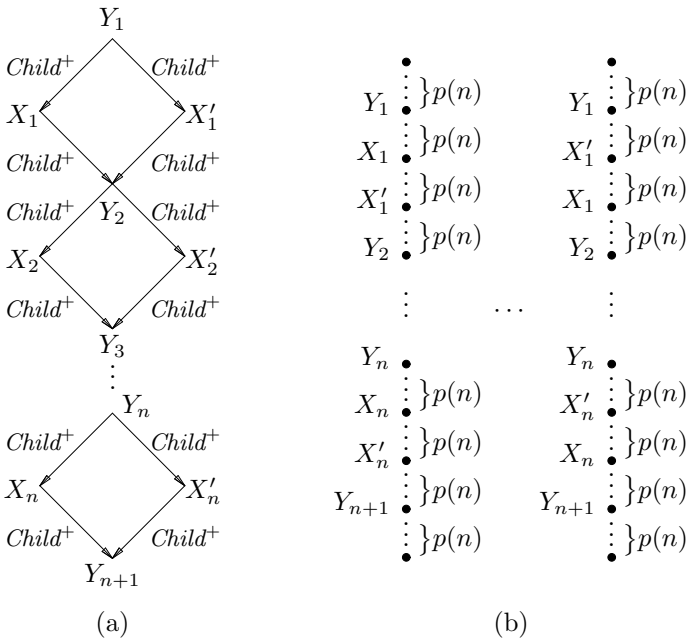



Fig. 9. Query $D_n$ (a) and path structures $PS(n, p(n))$ (b).

LEMMA 7.3. *Let $Q$ and $Q'$ be two $DABCQ[\{Child^*, Child^+\}]$ and $\Gamma$ be a set of labels. If there is a label-path in $LP(\Pi_{Q'})$ in which all labels from $\Gamma$ occur, but there is no label-path in $LP(\Pi_Q)$ in which all labels from $\Gamma$ occur, then there is a path-structure $\mathcal{M}$ on which $Q$ is true but $Q'$ is not.*

Now we can prove our theorem. The proofs of the two lemmata follow at the end of the section.

PROOF OF THEOREM 7.1. By contradiction. Assume there is a (Boolean) APQ $\mathcal{Q}$, that is, a finite union of ABCQs, which is equivalent to $D_n$, and that $\mathcal{Q}$ is of size bounded by polynomial $p(n)$. Let $s$ be a path of $p(n)$ unlabeled nodes. The regular expression

$$s.Y_1.s.(X_1.s.X'_1 \mid X'_1.s.X_1).s.Y_2.s.(X_2.s.X'_2 \mid X'_2.s.X_2).s.Y_3.s.\cdots . \\ s.Y_n.s.(X_n.s.X'_n \mid X'_n.s.X_n).s.Y_{n+1}.s$$

defines a set of $p(n)$-scattered path-structures over alphabet

$$\Sigma = \{X_1, \ldots, X_n, X'_1, \ldots, X'_n, Y_1, \ldots, Y_{n+1}\},$$

as sketched in Figure 9 (b). We refer to the set of these structures as $PS(n, p(n))$. It is easy to see that $D_n$ is true on each of the structures in $PS(n, p(n))$.

There are $2^n$ structures in $PS(n, p(n))$ and $D_n$ is true on all of them, but there are no more than $p(n)$ ABCQs in $\mathcal{Q}$. Therefore, there is an ABCQ $Q \in \mathcal{Q}$ which is true for at least $2^{n-\log p(n)}$ structures in $PS(n, p(n))$ and is not true on any structure on which $D_n$ is not true.

As the path structures in $PS(n, p(n))$ are $(p(n) \geq |Q|)$-scattered, by Lemma 7.2,

there is an $ABCQ[\{\mathit{Child}^+, \mathit{Child}^*\}]$ $Q'$ with $|Q'| \leq |Q|$, $Q' \subseteq Q$, and which is true on all structures in $PS(n, p(n))$ on which $Q$ is true.

In each path-structure $\mathcal{A}$ of $PS(n, p(n))$, for any $1 \leq j \leq n$, precisely one node $v$ is labeled $X_j$ and precisely one *different* node $w$ is labeled $X'_j$. Thus if query $Q'$ contains unary atoms $X_j(x_j)$ and $X'_j(x'_j)$, a mapping $\theta$ can only be a satisfaction of $Q'$ on $\mathcal{A}$ if $\theta(x_j) = v$ and $\theta(x'_j) = w$. But if there is a variable-path in $\Pi_{Q'}$ with $x_j$ above $x'_j$ (respectively, $x'_j$ above $x_j$) and $w$ above $v$ (respectively, $v$ above $w$) in $\mathcal{A}$, no satisfaction of $Q'$ on the structure can exist. Let $i_1, \ldots, i_m$ be *precisely* those pairwise distinct indexes for which, for $1 \leq j \leq m$, $Q'$ does not contain a variable-path containing two variables $x_{i_j}$ and $x'_{i_j}$ such that $X_{i_j}(x_{i_j})$ and $X'_{i_j}(x'_{i_j})$ are unary atoms of $Q'$. Then $Q'$ is true on at most $2^m$ path-structures of $PS(n, p(n))$.

We assumed that $Q$ is true on at least $2^{n-\log p(n)}$ structures of $PS(n, p(n))$ and showed that $Q'$ is true on the same. But then $m \geq n - \log p(n)$.

Since the (undirected) query graph of $Q'$ is a forest, the number of paths in $\Pi_{Q'}$ is not greater than the square of the number of its variables. As $|Q'| \leq p(n)$, $|\Pi_{Q'}| \leq p(n)^2$.

Now, if $n > 3 \cdot \log p(n)$, then $m > 2 \cdot \log p(n)$ and there are more choices

$$\Gamma = \{E_1 \in \{X_{i_1}, X'_{i_1}\}, \ldots, E_m \in \{X_{i_m}, X'_{i_m}\}\}$$

than there are paths in $\Pi_{Q'}$. Assume there are two *distinct* such choices $\Gamma, \Gamma'$ and a variable-path $\pi \in \Pi_{Q'}$ such that all labels of $\Gamma \cup \Gamma'$ occur in $\pi$. Then there is an index $i_j$ such that $X_{i_j}, X'_{i_j} \in \Gamma \cup \Gamma'$. This is in contradiction to the assumptions we made about the indexes $i_1, \ldots, i_m$. Thus there must be (at least) one such choice $\Gamma$ such that no single path in $\Pi_{Q'}$ exists in which all the labels of $\Gamma$ occur. Since there is a path in $D_n$ which contains all the labels of $\Gamma$, by Lemma 7.3, there is a model $\mathcal{M}$ of $Q'$ which is not a model of $D_n$. Since $Q' \subseteq Q$, $\mathcal{M}$ is also a model of $Q$.

This is in contradiction with our assumption that $Q \subseteq D_n$. Consequently, for $n > 3 \cdot \log p(n)$, there cannot be an ABCQ of size bounded by polynomial $p(n)$ that is contained in $D_n$ and is true on exponentially many structures of $PS(n, p(n))$. It follows that for sufficiently large $n$ there cannot be an APQ equivalent to $D_n$ that is of polynomial size. □

Now it remains to prove the two technical lemmata we used in the proof of our succinctness result.

### Proof of Lemma 7.2

We say that a Boolean query $Q'$ is a *faithful simplification* of a Boolean query $Q$ *w.r.t. a class of structures* $\mathbf{A}$ if $|Q'| \leq |Q|$, $Q' \subseteq Q$, and $Q'$ is true on structures of $\mathbf{A}$ on which $Q$ is true.

Below, by $G_Q$, we refer to the directed graph obtained from the query graph of $Q$ by removing all edges besides the *Child* edges. We will in particular consider the connected components of this graph, subsequently called the $G_Q$-components. We say that $C_0$ is a *parent component* of connected component $C$ of such a graph iff there is a variable $x$ in $C_0$ and a variable $y$ in $C$ such that there is an atom $\mathit{Child}^+(x, y)$ or $\mathit{Child}^*(x, y)$ in $Q$. (Of course, $C \neq C_0$ because the query graph of $Q$ is a forest.) The ancestors of a component are obtained by upward reachability through the parent relation on $G_Q$-components.

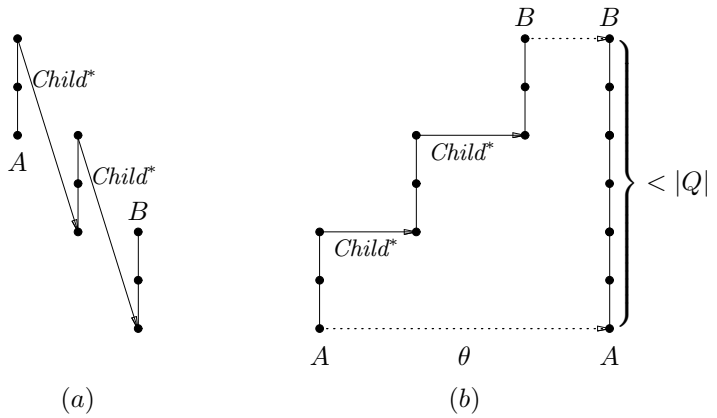



Fig. 10. A query (a) and the query embedded into a path structure with $B$ as high above $A$ as possible (b). The unlabeled edges are child-edges.

Lemma 7.2 immediately follows from the following four lemmata.

LEMMA 7.4. *Let $Q$ be an $ABCQ[\mathbf{Ax}]$ that is true on at least one path structure. Then there is an $ABCQ[\{Child, Child^*, Child^+\}]$ $Q'$ that is a faithful simplification of $Q$ w.r.t. the path-structures and in which each $G_{Q'}$-component is a path.*

PROOF. Query $Q$ cannot contain a *NextSibling*, *NextSibling*$^+$, or *Following*-atom, because if it does, $Q$ is false on all path-structures, contradicting our assumption that $Q$ is true on at least one path-structure.

Let $Q'$ be the query obtained from $Q$ by iteratively applying the following three rules until a fixpoint is reached.

—if $Q'$ contains atom $NextSibling^*(x, y)$, remove it and substitute all occurrences of variable $y$ in $Q'$ by $x$;

—if there are atoms $Child(x, z)$, $Child(y, z)$ in $Q'$, remove $Child(y, z)$ and substitute every occurrence of $y$ in $Q'$ by $x$;

—if there are atoms $Child(x, y)$, $Child(x, z)$ in $Q'$, remove $Child(x, z)$ and substitute every occurrence of $z$ in $Q'$ by $y$.

It is easy to verify that $Q'$ is a faithful simplification of $Q$ w.r.t. the path structures. Moreover, none of the rewrite rules can introduce a cycle into $Q'$, thus it is an $ABCQ[\{Child, Child^*, Child^+\}]$. There are neither atoms $Child(x, y)$ and $Child(x, z)$ with $y \neq z$ nor atoms $Child(x, z)$ and $Child(y, z)$ with $x \neq y$ in $Q'$, so each $G_{Q'}$-component is a path. □

Since each $G_Q$-component is a path, we may give the $k$ variables inside $G_Q$-component $C$ the names $x_1^C, \ldots, x_k^C$. We use $|C|$ to denote the number of variables $k$ in $G_Q$-component $C$. We will think of the node names of a path structure as an initial segment of the integers, thus $v > w$ if and only if $v$ is below $w$ in the path.

LEMMA 7.5. *Let $Q$ be an $ABCQ[\{Child, Child^*, Child^+\}]$ in which each $G_Q$-component is a path and that is true on at least one $|Q|$-scattered path structure $\mathcal{A}$. Then,*

*(a) any $G_Q$-component contains at most one label atom and*

*(b) if $C_1, \ldots, C_m$ is a path of $G_Q$-components and $Q$ contains unary atoms $L(x_k^{C_1})$ and $L'(x_l^{C_m})$ with $L \neq L'$, then the node labeled $L$ in $\mathcal{A}$ is above the node labeled $L'$.*

*(c) if $C_1, \ldots, C_m$ is a path of $G_Q$-components and $Q$ contains unary atoms $L(x_{k_1}^{C_1})$ and $L(x_{k_m}^{C_m})$, then for no $1 \leq j \leq m$ and $L' \neq L$ there can be a unary atom $L'(x_{k_j}^{C_j})$ in $Q$.*

PROOF. (a) Assume that there is a $G_Q$-component $C$ with two label atoms,

$$\mathit{Child}(x_1^C, x_2^C), \ldots, \mathit{Child}(x_{|C|-1}^C, x_{|C|}^C), L(x_k^C), L'(x_l^C)$$

with either $L \neq L'$ or $k \neq l$, a $|Q|$-scattered path-structure $\mathcal{A}$, and a satisfaction $\theta$ of $Q$ on $\mathcal{A}$. Since $|C| \leq |Q| - 2$, $|\theta(x_k^C) - \theta(x_l^C)| < |Q| - 2$. However, $\mathcal{A}$ is a $|Q|$-scattered path-structure and thus cannot contain two labels on a subpath of length $|Q|$. Contradiction.

(b) Let $\theta$ be a satisfaction of $Q$ on $\mathcal{A}$. Assume that $\theta(x_k^{C_1}) > \theta(x_l^{C_m})$, i.e., the node labeled $L$ is below the node labeled $L'$ in $\mathcal{A}$. Then, for each $1 \leq i < m$, there is an atom $R(x_{j_i}^{C_i}, x_{j_{i+1}}^{C_{i+1}})$ in $Q$ with $R$ either $\mathit{Child}^+$ or $\mathit{Child}^*$. So $\theta(x_{j_i}^{C_i}) \leq \theta(x_{j_{i+1}}^{C_{i+1}})$ and consequently $\theta(x_1^{C_i}) \leq \theta(x_{|C_{i+1}|}^{C_{i+1}}) = \theta(x_1^{C_{i+1}}) + |C_{i+1}| - 1$. But then $\theta(x_k^{C_1}) - \theta(x_l^{C_m}) \leq \Sigma_{i=1}^m (|C_i| - 1) < |Q|$. (See Figure 10 for an illustration.) This is in contradiction with our assumption that $\mathcal{A}$ is a $|Q|$-scattered path structure and thus $|\theta(x_k^{C_1}) - \theta(x_l^{C_m})| \geq |Q|$.

(c) follows immediately from (b) and the fact that in a $|Q|$-scattered path structure each label occurs at at most one node. □

Below, we will call the $G_Q$-components of an $\mathit{ABCQ}[\{\mathit{Child}, \mathit{Child}^*, \mathit{Child}^+\}]$ $Q$ *successor-repellent* if for any two atoms $R(x, y), R'(x', y')$ in $Q$ with $x = x', y \neq y'$ or $x \neq x', y = y'$, neither $R = \mathit{Child}$ nor $R' = \mathit{Child}$. The naming of this term is due to the following fact: Let $Q$ be a successor-repellent $\mathit{ABCQ}[\{\mathit{Child}, \mathit{Child}^*, \mathit{Child}^+\}]$. Then for any two components $C, C'$ such that $C'$ is a successor of $C$ and for any satisfaction $\theta$ of $Q$ (on a path structure), $\theta(x_{|C|}^C) \leq \theta(x_1^{C'})$.

LEMMA 7.6. *Let $Q$ be an $\mathit{ABCQ}[\{\mathit{Child}, \mathit{Child}^*, \mathit{Child}^+\}]$ that is true on at least one $|Q|$-scattered path structure and in which each $G_Q$-component is a path. Then there is an $\mathit{ABCQ}[\{\mathit{Child}, \mathit{Child}^*, \mathit{Child}^+\}]$ $Q'$ that is a faithful simplification of $Q$ w.r.t. the $|Q|$-scattered path-structures and whose $G_{Q'}$-components are successor-repellent.*

PROOF. We construct the query $Q'$ as follows. Initially, let $Q' := Q$. As often as possible, for any path of $G_Q$-components $C_1, \ldots, C_m$ such that there are atoms

$$R_1(x_{j_1}^{C_1}, x_{j'_1}^{C_2}), \ldots R_{m-1}(x_{j_{m-1}}^{C_{m-1}}, x_{j'_{m-1}}^{C_m}), L(x_k^{C_1}), L(x_l^{C_m})$$

in $Q'$ with $R_1, \ldots R_{m-1} \in \{\mathit{Child}^+, \mathit{Child}^*\}$, but there is no label atom over components $C_2, \ldots, C_{m-1}$, replace all occurrences of variable $x_l^{C_m}$ in $Q'$ by $x_k^{C_1}$ and delete atom $R_{m-1}(x_{j_{m-1}}^{C_{m-1}}, x_{j'_{m-1}}^{C_m})$. Moreover, for each $1 \leq i < m - 1$, if $R_i = \mathit{Child}^*$, remove the atom $\mathit{Child}^*(x_{j_i}^{C_i}, x_{j'_i}^{C_{i+1}})$ and substitute $x_{j'_i}^{C_{i+1}}$ by $x_{j_i}^{C_i}$ and if $R_i = \mathit{Child}^+$,

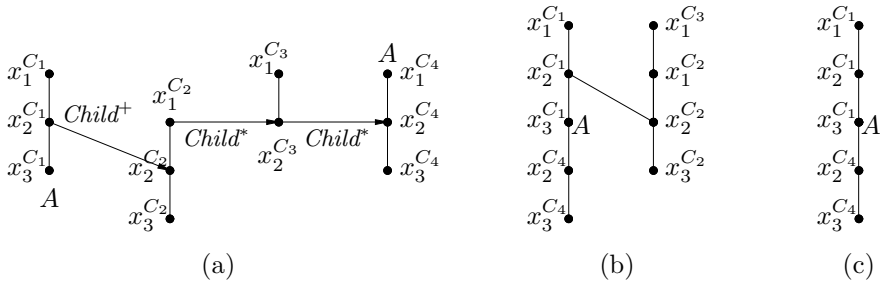



Fig. 11. A path of components with two occurrences of the same label (a), the same query after atom replacement and variable substitution (b), and (c) the query after applying the algorithm of Lemma 7.4. The unlabeled edges are child-edges.

replace $R_i$ by *Child*. Note that this query is an ABCQ. Then apply the algorithm of Lemma 7.4 to turn the $G_{Q'}$-components into paths. (See Figure 11 for an example of this construction.) To conclude with our construction, we replace each atom $R(x_k^{C_i}, x_l^{C_j})$ of $Q'$, where $R$ is either *Child*$^+$ or *Child*$^*$, by $R(x_{|C_i|}^{C_i}, x_1^{C_j})$.

Clearly, $Q'$ is a successor-repellent ABCQ with $|Q'| \leq |Q|$. Since $Q$ is true on at least one $|Q|$-scattered path structure, it follows from Lemma 7.5 (a) and (c) that there are no two $G_{Q'}$-components $C_1, C_m$ such that both are labeled $L$ and $C_1$ is an ancestor of $C_m$.

It is also easy to verify that $Q' \subseteq Q$: Given an arbitrary tree structure $\mathcal{A}$ and a satisfaction $\theta'$ for $Q'$ on $\mathcal{A}$, we can construct a satisfaction $\theta$ of $Q$ as $x \mapsto \theta'(y)$ if the construction of $Q'$ from $Q$ substituted $x$ by $y$ and $x \mapsto \theta'(x)$ otherwise. That $\theta$ is a satisfaction is obvious for all atoms of $Q$ apart from atom $R_{m-1}(x_{j_{m-1}}^{C_{m-1}}, x_{j'_{m-1}}^{C_m})$, which we simply deleted. But it is not hard to convince oneself that if $R_{m-1}(\theta(x_{j_{m-1}}^{C_{m-1}}), \theta(x_{j'_{m-1}}^{C_m}))$ does not hold, then it must be true that for every satisfaction $\theta_0$ of $Q$ on $\mathcal{A}$, *Child*$^*(\theta(x_{j_{m-1}}^{C_{m-1}}), \theta_0(x_{j_{m-1}}^{C_{m-1}}))$ and therefore *Child*$^+(\theta(x_{j'_{m-1}}^{C_m}), \theta_0(x_{j'_{m-1}}^{C_m}))$. But then, $\theta_0(x_k^{C_1}) \neq \theta_0(x_l^{C_m})$ and $\mathcal{A}$ must have at least two distinct nodes labeled $L$. This is in contradiction with our assumption that $Q$ is true on at least one $|Q|$-scattered path structure.

Moreover, $Q'$ is true on all $|Q|$-scattered path-structures on which $Q$ is true. To show this, let $Q$ be true on some $|Q|$-scattered path-structure $\mathcal{A}$ with satisfaction $\theta$. We construct a satisfaction $\theta'$ for $Q'$ on $\mathcal{A}$ using the following algorithm.

1 for each $G_{Q'}$-component $C_j$ do
// process components according to some topological ordering w.r.t.
// *Child*$^+$, *Child*$^*$: if there is an atom *Child*$^+(x_k^{C_i}, x_l^{C_j})$ or *Child*$^+(x_k^{C_i}, x_l^{C_j})$
// in $Q'$, then $\theta'(x_1^{C_i}), \ldots, \theta'(x_{|C_i|}^{C_i})$ has been computed before.
2 begin
3 if $Q'$ contains a label atom $L(x_l^{C_j})$ then
4 $\theta'(x_1^{C_j}) := \theta(x_l^{C_j}) - l + 1$;
// $\theta(x_l^{C_j})$ is the unique node of the path structure which has label $L$;
5 else $\theta'(x_1^{C_j}) := 1 + \max\{\theta'(x_{|C_i|}^{C_i}) \mid C_i$ is a parent $G_{Q'}$-component of $C_j\}$;

// let $\max(\emptyset) = 0$

6 for the remaining $1 < k \leq |C_j|$ do
7 $\theta'(x_k^{C_j}) := \theta'(x_1^{C_j}) + k - 1$;
8 end;

Clearly, this algorithm defines $\theta'$ for all variables of $Q'$. Since for any $x$, $\theta'(x)$ cannot be greater than $\max\{v \mid \text{path-structure node } v \text{ has a label}\} + |Q'|$, $\theta'$ maps into the ($|Q|$-scattered) path structure.

Lines 6-7 assure that all the *Child*-atoms of $Q'$ are true. Line 4 assures that the label-atoms are true: otherwise, $\theta$ could not be a satisfaction of $Q$. For a component $C_j$ without a label-atom, line 5 assures that all atoms of the form $R(x_{|C_i|}^{C_i}, x_1^{C_j})$, for $R$ either *Child*$^+$ or *Child*$^*$, are satisfied because $\theta'(x_1^{C_j}) = \theta'(x_{|C_i|}^{C_i}) + 1$.

Finally, lines 3-4 handle the case that component $C_j$ contains a label atom $L(x_l^{C_j})$. By Lemma 7.5 (a) the choice of label atom for the component in line 3 is deterministic. What has to be shown is that

$$\theta'(x_1^{C_j}) \geq 1 + \max\{\theta'(x_{|C_i|}^{C_i}) \mid C_i \text{ is a parent } G_{Q'}\text{-component of } C_j\}.$$

It is easy to verify by induction that

$$\max\{\theta'(x_{|C_i|}^{C_i}) \mid C_i \text{ is a parent } G_{Q'}\text{-component of } C_j\} < v + |Q| - |C_j|,$$

where $v$ is the bottommost among the nodes of the path structure carrying labels $L_0$ that appear in the ancestor components of $C_j$. Thus, if all these labels $L_0$ occur above $\theta(x_l^{C_j})$, we are done. (In a $|Q|$-scattered path structure, $|\theta(x_l^{C_j}) - v| \geq |Q|$.)

We know that by our construction, label $L$ does not occur in any of the ancestor-components of $C_j$. But then, if all labels that occur in ancestor-components of $C_j$ differ from $L$, by Lemma 7.5 (b) the path-structure node $v$ must be above $\theta'(x_1^{C_j})$, otherwise $\theta$ would not be a satisfaction of $Q$. □

LEMMA 7.7. *Let $Q$ be an ABCQ[{Child, Child$^*$, Child$^+$}] such that the components of $G_Q$ are successor-repellent and each $G_Q$-component contains at most one label-atom. Then, the query $Q'$ obtained from $Q$ by replacing each occurrence of predicate Child by Child$^+$ is equivalent to $Q$.*

PROOF. Since *Child* $\subseteq$ *Child*$^+$, it is obvious that $Q \subseteq Q'$. For the other direction, let $\theta'$ be any satisfaction of $Q'$. We define a valuation $\theta$ for $Q$ from $\theta'$. For every $G_Q$-component $C$, let $\theta(x_k^C) := \theta'(x_l^C) + k - l$ if there is a label-atom over variable $x_l^C$ – as shown above, there is at most one such variable per component – or $\theta(x_k^C) := \theta'(x_1^C) + k - 1$ if component $C$ does not contain a label-atom. It is now easy to verify that $\theta$ is indeed a satisfaction for $Q$: The label- and *Child*-atoms of $Q$ are satisfied by definition. Since $\theta(x_{|C_i|}^{C_i}) \leq \theta'(x_{|C_i|}^{C_i})$ and $\theta(x_1^{C_j}) \geq \theta'(x_1^{C_j})$, $R(\theta'(x_{|C_i|}^{C_i}), \theta'(x_1^{C_j}))$, where $R$ is either *Child*$^*$ or *Child*$^+$, implies $R(\theta(x_{|C_i|}^{C_i}), \theta(x_1^{C_j}))$. Thus, $Q' \subseteq Q$ and consequently $Q' \equiv Q$. □

### Proof of Lemma 7.3

PROOF OF LEMMA 7.3. We define a number of restrictions of the set $\Pi_Q$ of variable-paths in $Q$. For labels $X$, let $\Pi_Q|_X$ denote the set of variable-paths in $\Pi_Q$

which contain a variable with label $X$. Let $\Pi_Q|_{\neg X} = \Pi_Q - \Pi_Q|_X$ and $\Pi_Q|_{\phi\wedge\psi} = \Pi_Q|_\phi \cap \Pi_Q|_\psi$. For variables $x$, let $\Pi_Q|_x$ denote the set of all variable-paths in $\Pi_Q$ in which $x$ occurs. Let $LC(\psi)$ denote the label-paths in $LP(\Pi_Q|_\psi)$ concatenated in any (say, lexicographic) order.

Let $\Gamma = \{E_1, \ldots, E_m\}$. There is a variable-path $x_1.\cdots.x_k \in \Pi_{Q'}$ and query $Q'$ contains atoms $E_1(x_{i_1}), \ldots, E_m(x_{i_m})$ such that, w.l.o.g., $1 \le i_1 \le \cdots \le i_m \le k$. By assumption, there is no such variable-path in $\Pi_Q$.

*Construction of path-structure $\mathcal{M}$.* We define $\mathcal{M}$ as the path structure

$$LC(\neg E_1).LC(E_1 \wedge \neg E_2).LC(E_1 \wedge E_2 \wedge \neg E_3).\cdots.LC(E_1 \wedge \cdots \wedge E_{m-1} \wedge \neg E_m)$$

Since $\Pi_Q|_{E_1\wedge\cdots\wedge E_m}$ is empty, $\mathcal{M}$ is a concatenation of *all* paths in $LP(\Pi_Q)$.

*$\mathcal{M}$ is a model of $Q$.* We show that $Q$ is true on any concatenation of the label-paths of $LP(\Pi_Q)$. Consider the partial function $\theta$ from variables of $Q$ to nodes in $\mathcal{M}$ defined as

$$\theta(x) = v \;\Leftrightarrow\; v \text{ is the topmost node in } \mathcal{M} \text{ such that for all } \pi.x.\pi' \in \Pi_Q|_x,\\ \pi.x \text{ can be matched in the path from the root of } \mathcal{M} \text{ to } v.$$

We say that a variable-path $\pi \in \Pi_Q$ can be matched in a subpath $\pi'$ of a path structure iff each of the variables $x$ in $\pi$ can be mapped to a node $\alpha(x)$ in $\pi'$ such that if $L(x)$ is an atom in $Q$, $\alpha(x)$ carries label $L$, and if $x$ occurs before $y$ in $\pi$, $\alpha(x)$ occurs before $\alpha(y)$ in $\pi'$.

As $\mathcal{M}$ is a concatenation of all paths in $LP(\Pi_Q)$, for each $x$, the label-paths of all prefixes of paths in $\Pi_Q(x)$ occur in $\mathcal{M}$. Thus $\theta$ is defined for all variables in $Q$.

The valuation $\theta$ is also consistent. By definition, $\theta$ satisfies all unary ("label") atoms. Consider a binary atom $\mathit{Child}^+(x,y)$ or $\mathit{Child}^*(x,y)$. (Thus there is a path $\pi.x.y.\pi' \in \Pi_Q$.) Assume that $\theta(x) = v$ and $\theta(y) = w$. By definition, $v$ is the topmost node such that all variable-paths with a prefix $\pi_0.x$ can be matched in the subpath of $\mathcal{M}$ from the root to $v$. For each such $\pi_0.x$, $\pi_0.x.y$ must match the path from the root of $\mathcal{M}$ to $w$. Thus, $w$ must be below $v$ in $\mathcal{M}$.

*$\mathcal{M}$ is not a model of $Q'$.* Assume there is a satisfaction $\theta$ of $Q'$ on $\mathcal{M}$.

(1) By definition, $\theta(x_{i_1})$ cannot be a node in $LC(\neg E_1)$.

($j \to j+1$) Induction step: Assume that $\theta(x_{i_j})$ cannot be a node in the prefix $LC(\neg E_1).\cdots.LC(E_1\wedge\cdots\wedge E_{j-1}\wedge\neg E_j)$ of $\mathcal{M}$. For $\theta$ to be a satisfaction, $\theta(x_{i_{j+1}})$ must either be a descendant of $\theta(x_{i_j})$ or $\theta(x_{i_{j+1}}) = \theta(x_{i_j})$. By the induction hypothesis, $\theta(x_{i_{j+1}})$ cannot be in $LC(\neg E_1).\cdots.LC(E_1 \wedge \cdots \wedge E_{j-1} \wedge \neg E_j)$. But by definition $\theta(x_{i_{j+1}})$ cannot be a node in $LC(E_1 \wedge \cdots \wedge E_j \wedge \neg E_{j+1})$ either. It follows that $\theta(x_{i_{j+1}})$ cannot be a node in $LC(\neg E_1).\cdots.LC(E_1\wedge\cdots\wedge E_j\wedge\neg E_{j+1})$.

So $\theta(x_{i_m})$ must remain undefined. Contradiction with our assumption that $\theta$ is a satisfaction of $Q'$ on $\mathcal{M}$. □

We illustrate the construction by an example.

Example 7.8. Consider the 2-diamond query $D_2$ shown in Figure 12 (a) and the ABCQ $Q$ of Figure 12 (b). In $Q$ there is no path that contains both $E_1 = X_1'$ and $E_2 = X_2'$, while $D_2$ contains such a path. The path-structure $\mathcal{M} =$

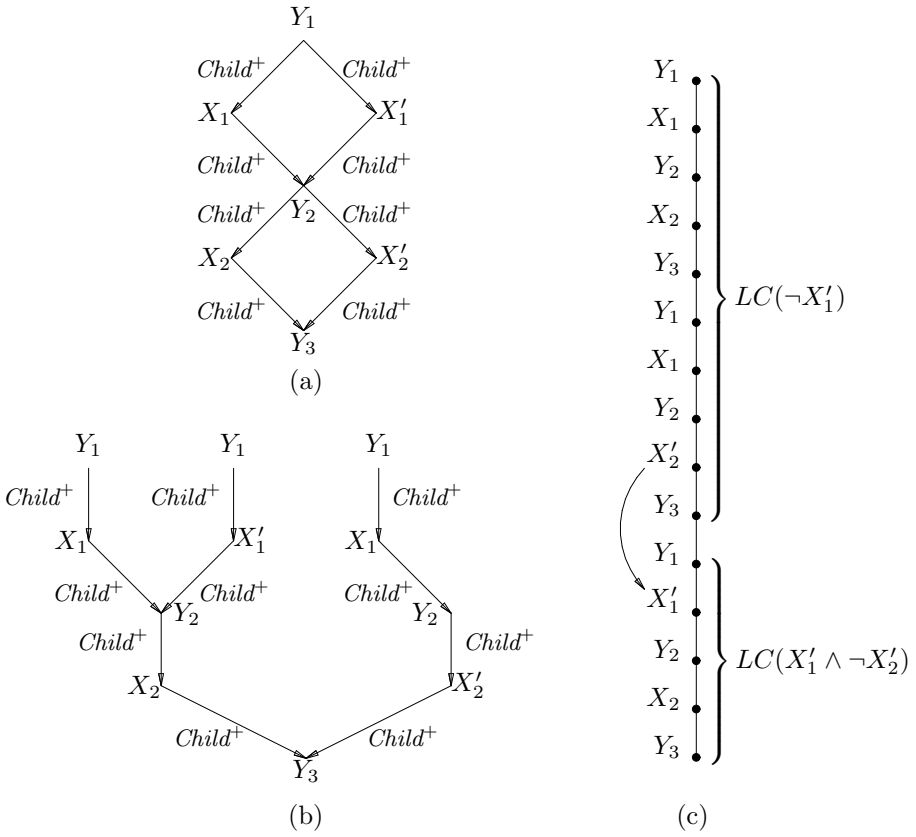



Fig. 12. Example of the path structure construction of the proof of Lemma 7.3.

$LC(\neg X_1').LC(X_1' \wedge \neg X_2')$ constructed as described above is shown in Figure 12 (c). It consists of a concatenation of the two paths $Y_1.X_1.Y_2.X_2.Y_3$ and $Y_1.X_1.Y_2.X_2'.Y_3$ – which do not contain $X_1'$ (and which we can add to $\mathcal{M}$ in any order) – with the path $Y_1.X_1'.Y_2.X_2.Y_3$, which contains $X_1'$ but not $X_2'$ and which is therefore appended to $\mathcal{M}$ after the other two paths. It is easy to see that indeed $Q$ is true on $\mathcal{M}$. However, $D_2$ is false on $\mathcal{M}$. (The unique occurrence of $X_1'$ in $\mathcal{M}$ is a descendant of the unique occurrence of $X_2'$.) This witnesses that $Q \not\subseteq D_2$. □

## REFERENCES


Abiteboul, S., Hull, R., and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Baumgartner, R., Flesca, S., and Gottlob, G. 2001. "Visual Web Information Extraction with Lixto". In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*. Rome, Italy, 119–128.

Benedikt, M., Fan, W., and Kuper, G. 2003. "Structural Properties of XPath Fragments". In *Proc. of the 9th International Conference on Database Theory (ICDT)*. Siena, Italy, 79–95.

Bodirsky, M., Duchier, D., Niehren, J., and Miele, S. 2004. "A New Algorithm for Normal Dominance Constraints". In *Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. New Orleans, Louisiana, USA, 59–67.

Chandra, A. K. and Merlin, P. M. 1977. "Optimal Implementation of Conjunctive Queries

in Relational Data Bases". In *Conference Record of the Ninth Annual ACM Symposium on Theory of Computing (STOC'77)*. Boulder, CO, USA, 77–90.

Chekuri, C. and Rajaraman, A. 1997. Conjunctive Query Containment Revisited". In *Proc. of the 6th International Conference on Database Theory (ICDT)*. Delphi, Greece, 56–70.

Dechter, R. 2003. *"Constraint Processing"*. Morgan Kaufmann.

Deutsch, A. and Tannen, V. 2003a. "MARS: A System for Publishing XML from Mixed and Redundant Storage". In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)*. Berlin, Germany, 201–212.

Deutsch, A. and Tannen, V. 2003b. "Reformulation of XML Queries and Constraints". In *Proc. of the 9th International Conference on Database Theory (ICDT)*. 225–241.

Ebbinghaus, H.-D. and Flum, J. 1999. *Finite Model Theory*. Springer-Verlag. Second edition.

Flum, J., Frick, M., and Grohe, M. 2002. "Query Evaluation via Tree-Decompositions". *Journal of the ACM* **49**, 6, 716–752.

Gottlob, G. and Koch, C. 2002. "Monadic Queries over Tree-Structured Data". In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science (LICS)*. Copenhagen, Denmark, 189–202.

Gottlob, G. and Koch, C. 2004. "Monadic Datalog and the Expressive Power of Web Information Extraction Languages". *Journal of the ACM* **51**, 1, 74–113.

Gottlob, G., Koch, C., and Pichler, R. 2005. "Efficient Algorithms for Processing XPath Queries". *ACM Transactions on Database Systems* **30**, 2 (June), 444–491.

Gottlob, G., Koch, C., Pichler, R., and Segoufin, L. 2005. "The Complexity of XPath Query Evaluation and XML Typing". *Journal of the ACM* **52**, 2 (Mar.), 284–335.

Gottlob, G., Koch, C., and Schulz, K. U. 2004. "Conjunctive Queries over Trees". In *Proceedings of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'04)*. Paris, France, 189–200.

Gutjahr, W., Welzl, E., and Woeginger, G. 1992. "Polynomial Graph Colourings". *Discrete Applied Math.* **35**, 29–46.

Hell, P. and Nesetril, J. 2004. *Graphs and Homomorphisms*. Oxford University Press.

Hell, P., Nesetril, J., and Zhu, X. 1996a. "Complexity of Tree Homomorphisms". *Discrete Applied Mathematics* **70**, 1, 23–36.

Hell, P., Nesetril, J., and Zhu, X. 1996b. "Duality and Polynomial Testing of Tree Homomorphisms". *Transactions of the American Mathematical Society* **348**, 4 (Apr.), 1281–1297.

Hidders, J. 2003. "Satisfiability of XPath Expressions". In *Proc. 9th International Workshop on Database Programming Languages (DBPL)*. Potsdam, Germany, 21–36.

Kolaitis, P. and Vardi, M. 1998. "Conjunctive-Query Containment and Constraint Satisfaction". In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'98)*. 205–213.

LDC. 1999. "The Penn Treebank Project". http://www.cis.upenn.edu/∼treebank/home.html.

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

Maier, D. 1983. *The Theory of Relational Databases*. Computer Science Press.

Marcus, M. P., Hindle, D., and Fleck, M. M. 1983. "D-Theory: Talking about Talking about Trees". In *Proc. 21st Annual Meeting of the Association for Computational Linguistics (ACL)*. 129–136.

Marx, M. 2005. "First Order Paths in Ordered Trees". In *Proc. ICDT 2005*. 114–128.

Meuss, H. and Schulz, K. U. 2001. "Complete Answer Aggregates for Tree-like Databases: A Novel Approach to Combine Querying and Navigation". *ACM Transactions on Information Systems* **19**, 2, 161–215.

Meuss, H., Schulz, K. U., and Bry, F. 2001. "Towards Aggregated Answers for Semistructured Data". In *Proc. of the 8th International Conference on Database Theory (ICDT'01)*. 346–360.

Minoux, M. 1988. "LTUR: A Simplified Linear-Time Unit Resolution Algorithm for Horn Formulae and Computer Implementation". *Information Processing Letters* **29**, 1, 1–12.

Olteanu, D., Meuss, H., Furche, T., and Bry, F. 2002. "XPath: Looking Forward". In *Proc. EDBT Workshop on XML Data Management*. Vol. LNCS 2490. Springer-Verlag, Prague, Czech Republic, 109–127.

Schaefer, T. 1978. "The Complexity of Satisfiability Problems". In *Proc. 10th Ann. ACM Symp. on Theory of Computing (STOC)*. 216–226.

Schmidt-Schauss, M. and Schulz, K. U. 1998. "On the Exponent of Periodicity of Minimal Solutions of Context Equations". In *Proc. 9th Int. Conf. on Rewriting Techniques and Applications*. 61–75.

Schmidt-Schauss, M. and Schulz, K. U. 2002. "Solvability of Context Equations with Two Context Variables is Decidable". *Journal of Symbolic Computation* **33**, 1, 77–122.

Schmidt-Schauss, M. and Stuber, J. 2001. "On the Complexity of Linear and Stratified Context Matching Problems". Unpublished manuscript.

Schwentick, T. 2000. "On Diving in Trees". In *Proc. International Symposium on Mathematical Foundations of Computer Science (MFCS)*. 660–669.

Vardi, M. Y. 1982. "The Complexity of Relational Query Languages". In *Proc. 14th Annual ACM Symposium on Theory of Computing (STOC'82)*. San Francisco, CA USA, 137–146.

World Wide Web Consortium. 1999. XML Path Language (XPath) Recommendation. http://www.w3c.org/TR/xpath/.

Yannakakis, M. 1981. "Algorithms for Acyclic Database Schemes". In *Proceedings of the 7th International Conference on Very Large Data Bases (VLDB'81)*. Cannes, France, 82–94.