Decidable Containment of Recursive Queries

Diego Calvanese [*,1] Giuseppe De Giacomo [1]

Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza",
Via Salaria 113, 00198 Roma, Italy

Moshe Y. Vardi [2]

Department of Computer Science
Rice University, P.O. Box 1892
Houston, TX 77251-1892, U.S.A.

Abstract

One of the most important reasoning tasks on queries is checking containment, i.e., verifying whether one query yields necessarily a subset of the result of another one. Query containment is crucial in several contexts, such as query optimization, query reformulation, knowledge-base verification, information integration, integrity checking, and cooperative answering. Containment is undecidable in general for Datalog, the fundamental language for expressing recursive queries. On the other hand, it is known that containment between monadic Datalog queries and between Datalog queries and unions of conjunctive queries are decidable. It is also known that containment between unions of conjunctive two-way regular path queries, which are queries used in the context of semistructured data models containing a limited form of recursion in the form of transitive closure, is decidable. In this paper we combine the automata-theoretic techniques at the base of these two decidability results to show that containment of Datalog in union of conjunctive two-way regular path queries is decidable in 2EXPTIME. By sharpening a known lower bound result for containment of Datalog in union of conjunctive queries we show also a matching lower bound.

Key words: query containment, semistructured data, Datalog, regular path queries

1 Introduction

Querying is the fundamental mechanism for extracting information from a database. The basic reasoning task associated to querying is query answering, which amounts to computing the information to be returned as result of a query. There are, however, other reasoning services involving queries that data and knowledge representation systems should support. One of the most important is checking containment, i.e., verifying whether one query yields necessarily a subset of the result of another one. Query containment, called *subsumption* in AI [1,2], is crucial in several contexts, such as query optimization, query reformulation, knowledge-base verification, information integration, integrity checking, and cooperative answering; cf. [3–13]. Thus, it is fair to describe query containment as one of the most fundamental database reasoning tasks.

Needless to say, query containment is undecidable if we do not limit the expressive power of the query language; it is clearly undecidable for first-order logic. In fact, in knowledge representation suitable query languages have been designed for retaining decidability. The same is true in databases, where the notion of *conjunctive query* is the basic one in the investigation of reasoning about queries [14]. A conjunctive query (CQ) is simply a conjunction of atoms, where each atom is built out from relation symbols and (existentially quantified) variables. Relationally, a CQ is a project-join query. By adding union and recursion to conjunctive queries, one gets *Datalog*, the language of logic programs (known also as Horn-clause programs) without function symbols [15], which is essentially a fragment of fixpoint logic [16,17]. Datalog consists, in a pure way, only of the most fundamental elements of relational queries: join, projection, union, and recursion. With respect to query containment, CQs and Datalog span the spectrum in terms of computational complexity. In [14] it is shown that CQ containment is equivalent to CQ evaluation (NP-complete). (For some extensions, see [18–21].) On the other hand, it is shown in [22] that containment of Datalog queries is undecidable; the proof is by reduction from

* Corresponding author
Email addresses: calvanese@dis.uniroma1.it (Diego Calvanese), degiacomo@dis.uniroma1.it (Giuseppe De Giacomo), vardi@cs.rice.edu (Moshe Y. Vardi).
URLs: http://www.dis.uniroma1.it/calvanese/ (Diego Calvanese), http://www.dis.uniroma1.it/~degiacomo/ (Giuseppe De Giacomo), http://www.cs.rice.edu/~vardi/ (Moshe Y. Vardi).
[1] Supported in part by EU Project INFOMIX (Boosting Information Integration) IST-2001-33570, and by EU Project SEWASIE (Semantic Webs and AgentS in Integrated Economies) IST-2001-34825.
[2] Supported in part by NSF grants CCR-9988322, CCR-0124077, IIS-9908435, IIS-9978135, and EIA-0086264.

the containment problem for context-free grammars.

The most powerful query-containment results for Datalog are given in [23–25]. In [23] it is pointed out that tree-automata techniques can be used to prove the decidability of query containment for *monadic* Datalog, where rule heads use a single variable (which means that intermediate result of the query, as well as the final one, are sets of data elements). The other results apply to the relationship between Datalog and non-recursive Datalog (non-recursive Datalog queries are in essence unions of conjunctive queries). In [24] it is shown that checking containment of nonrecursive Datalog queries in Datalog queries is decidable in exponential time. In [25] (see also [21]) it is shown, using tree-automata techniques, that containment of Datalog queries in nonrecursive Datalog queries is decidable in triply exponential time. When the non-recursive query is represented, via unfolding, as a union of CQs, the complexity is doubly exponential, rather than triple exponential. (These bounds are known to be optimal, see [26,4] for studies of special cases and some extensions.)

In this paper we address the problem of query containment in the context of semistructured data models. Our goal is to capture the essential features found in databases, both traditional and semistructured, as well as knowledge bases in semantic networks, conceptual graphs, and description logics. For this purpose, we conceive a database as an edge-labeled graph, where nodes represent objects, and a labeled edge between two nodes represents the fact that the binary relation denoted by the label holds for the objects. This model captures data expressed using XML-like languages [27,28] and is accepted as a standard model for semistructured data [29,30].

In this framework, a basic querying mechanism is the one of *regular path queries* (RPQ) [29,31,32], which ask for all pairs of objects that are connected by a path conforming to a regular expression. Regular path queries are extremely useful for expressing complex navigations in a graph. In particular, union and transitive closure are crucial when we do not have a complete knowledge of the structure of the database. In our regular path queries, we include also the *inverse* operator, which enables us to navigate edges backwards [29,7], for example, from a child to its parent. We denote these queries by 2RPQs (two-way regular path queries). Using 2RPQs as the basic querying mechanism, one can construct *conjunctive 2-way regular path queries* (C2RPQs), which enables us to perform joins and projections over 2RPQs. C2RPQs are the basic building blocks for querying semistructured data [33,13,31]. The containment problem for C2RPQs (actually for unions of such C2RPQs) was studied in [34] (see also [33]), where it was shown, using two-way automata, to be EXPSPACE-complete.

The notable fact about the decidability of containment for C2RPQs is that C2RPQs are a fragment of recursive Datalog, due to the transitive closure op-

erator. Thus, the result in [33,34] is the first decidability result for containment of non-monadic recursive Datalog queries. The fact that automata-theoretic techniques are used both in [25] and in [34] suggests that perhaps the two decidability results can be combined. We show here that this is indeed the case by proving the decidability of the containment of Datalog queries in union of C2RPQs (which, implies the known decidability result for containment of union of C2RPQs). The automata-theoretic techniques combine tree automata with two-way automata; we use alternating two-way tree automata [35]. The upper bound is doubly exponential time, just as in [25], which we show to be optimal.

The rest of the paper is organized as follows. In Section 2 we present the data model and query languages for semistructured data we adopt in this paper. In Section 3 we provide some preliminary results on the characterization of containment of Datalog queries in unions of conjunctive queries. In Section 4 we introduce two-way alternating tree automata, which are used in Section 5 to establish the upper bound for containment of Datalog in unions of C2RPQs. In Section 6 we show a matching lower bound. Finally, in Section 7 we conclude the paper by discussing the impact of our results on view-based query processing.

2 Databases and Queries

We consider a *semistructured database* (DB) $\mathcal{G}$ as an edge-labeled graph $(\mathcal{D}, \mathcal{E})$, where $\mathcal{D}$ is the set of nodes, and $\mathcal{E}$ is the set of edges labeled with elements of an alphabet Δ. A node represents an object, and an edge between nodes d_1 and d_2 labeled e, denoted $e(d_1, d_2)$, represents the fact that the binary relation e holds for the pair (d_1, d_2).

The basic querying mechanism on a DB is that of *regular path queries* (RPQs). An RPQ E is expressed as a regular expression or a finite automaton, and computes the set of pairs of nodes of the DB connected by a path that conforms to the regular language $L(E)$ defined by E. We consider unions of conjunctive 2-way regular path queries [34], which extend regular path queries with the possibility to traverse edges backward, with conjunctions and variables, and with union.

Formally, Let Δ be a set of binary relation symbols, and let $\Delta^{\pm} = \Delta \cup \Delta^{-}$, with $\Delta^{-} = \{e^{-} \mid e \in \Delta\}$. Intuitively, e^{-} denotes the inverse of the binary relation e. If $r \in \Delta^{\pm}$, then we use r^{-} to mean the *inverse* of the relation r, i.e., if r is e, then r^{-} is e^{-}, and if r is e^{-}, then r^{-} is e.

2-way regular path queries (2RPQs) are expressed by means of regular expres-

sions or finite word automata over $\Delta^{\pm}$. Thus, in contrast with RPQs, 2RPQs may use also the inverse e^- of e, for each $e \in \Delta$. When evaluated over a DB $\mathcal{G}$, a 2RPQ E computes the set $E(\mathcal{G})$ of pairs of nodes (d_0, d_q) such that $r_1(d_0, d_1), r_2(d_1, d_2), \ldots, r_q(d_{q-1}, d_q)$ hold in $\mathcal{G}$ and $r_1 r_2 \cdots r_q$ is in the regular language $L(E)$ defined by E. Observe that, when $q = 0$, we have that $r_1 r_2 \cdots r_q = \varepsilon$ and $d_0 = d_q$.

Conjunctive 2-way regular path queries (C2RPQs) are conjunctions of atoms, where each atom specifies that one 2RPQ holds between two variables. More precisely a C2RPQ γ of *arity* n is a formula of the form

$$Q(x_1, \ldots, x_n) \leftarrow E_1(y_1, y_1'), \ldots, E_m(y_m, y_m')$$

where $x_1, \ldots, x_n, y_1, y_1' \ldots, y_m, y_m'$ range over a set $\{u_1, ..., u_k\}$ of variables and $E_1, \ldots, E_m$ are 2RPQs. The variables $x_1, \ldots, x_n$ are called *distinguished variables*. The *answer set* $\gamma(\mathcal{G})$ to a C2RPQ γ over a DB $\mathcal{G} = (\mathcal{D}, \mathcal{E})$ is the set of tuples $(d_1, \ldots, d_n)$ of nodes of $\mathcal{G}$ such that there is a total mapping σ from $\{u_1, \ldots, u_k\}$ to $\mathcal{D}$ with $\sigma(x_i) = d_i$ for every distinguished variable x_i of γ, and $(\sigma(y), \sigma(y')) \in E(\mathcal{G})$ for every conjunct $E(y, y')$ in γ. When the arity of γ is 0, then it is viewed as a Boolean query; the answer set is either the empty set (corresponding to *false*) or the set containing the 0-ary tuples (corresponding to *true*).

Finally, a *union of conjunctive 2-way regular path queries* of arity n has the form $\cup_i \gamma_i$, where each γ_i is a C2RPQ of arity n. The answer set to a union of C2RPQs $\Gamma = \cup_i \gamma_i$ over a DB $\mathcal{G}$ is simply $\Gamma(\mathcal{G}) = \cup_i \gamma_i(\mathcal{G})$. Notice that traditional *conjunctive queries* (resp., unions of conjunctive queries) (cf. [15]) are just a special case of C2RPQs (resp., unions of C2RPQs) in which each 2RPQ in an atom is simply a relation symbol.

A Datalog program consists of a set of Horn rules. A *(Horn) rule* is a first-order material implication between a body and a head, where the head consists of a single atom, and the body consists of a conjunction of atoms. Each atom is a formula of the form $R(x_1, \ldots, x_n)$ where R is a predicate symbol and $x_1, \ldots, x_n$ are variables. All variables are implicitly universally quantified outside the rule, and all variables appearing in the head are among the variables in the body. The predicates that occur in heads of rules are called *intensional* (IDB) predicates. The rest of the predicates are called *extensional* (EDB) predicates. Here we consider Datalog programs that are evaluated over a semistructured database. Hence, when not explicitly noted otherwise, we assume that the EDB predicates are among the predicates in Δ, which are all binary. Observe, however, that IDB predicates, which are not in Δ, may be of arbitrary arity.

We define now the answer set to a Datalog program Π with goal predicate

Q over a DB $\mathcal{G}$. Let $\mathcal{D}$ be a collection of facts about the extensional and intensional predicates of Π. Then, the facts that can be deduced from $\mathcal{D}$ by applying a rule

$$R(x_1, \ldots, x_n) \leftarrow R_1(\mathbf{y}^1), \ldots, R_m(\mathbf{y}^m)$$

of Π are all facts of the form $R(d_1, \ldots, d_n)$ such that $d_1, \ldots, d_n$ are nodes of Γ and there is a substitution of the variables in the body of the rule with nodes of Γ that substitutes x_i with d_i and such that, after the substitution, all atoms of the body are among the facts in $\mathcal{D}$. We denote by $\Theta_\Pi(\mathcal{D})$ the collection of facts obtained as the union of $\mathcal{D}$ with all facts that can be deduced from $\mathcal{D}$ by applying one of the rules of Π. For a DB $\mathcal{G}$, let

$$\begin{aligned} \mathcal{D}_0 &= \mathcal{G} \\ \mathcal{D}_{i+1} &= \Theta_\Pi(\mathcal{D}_i) \end{aligned}$$

Then, for an IDB predicate Q of Π, the *answer set* $Q^\infty_\Pi(\mathcal{G})$ to the Datalog program Π with goal predicate Q over the DB $\mathcal{G}$ is the collection of facts about Q in $\mathcal{D}_h$, where h is the least number such that $\mathcal{D}_h = \mathcal{D}_{h+1}$. Note that such an h always exists [15].

We say that a Datalog program Π with goal predicate Q is *contained* in a union of C2RPQs Γ if $Q^\infty_\Pi(\mathcal{G}) \subseteq \Gamma(\mathcal{G})$ for every database $\mathcal{G}$.

3 Containment of Datalog in Unions of Conjunctive Queries

A *containment mapping* from a conjunctive query ψ to a conjunctive query φ is a renaming of variables subject to the following constraints: (a) every distinguished variable must map to itself, and (b) after renaming, every literal in ψ must be among the literals of φ. It is well known that containment of conjunctive queries can be characterized in terms of containment mappings (cf. [15]). In fact this characterization has been extended in [19] to unions of conjunctive queries, and holds also for infinite unions.

Theorem 1 ([19]) *Let $\Phi = \cup_i \varphi_i$ and $\Psi = \cup_i \psi_i$ be (possibly infinite) unions of conjunctive queries. Then Φ is contained in Ψ (i.e., $\Phi(\mathcal{G}) \subseteq \Psi(\mathcal{G})$ for every database $\mathcal{G}$) if and only if for each φ_i there is a ψ_j such that φ_i is contained in ψ_j, i.e., there is a containment mapping from ψ_j to φ_i.*

As for containment of Datalog in (unions) of conjunctive queries, it is known (cf. [36,37]) that the relation defined by an IDB predicate Q in a Datalog

program Π, i.e., $Q_\Pi^\infty(\mathcal{G})$, can be defined by a possibly *infinite* union of conjunctive queries. That is, for each IDB predicate Q there is an infinite sequence $\varphi_0, \varphi_1, \ldots$ of conjunctive queries such that, for every database $\mathcal{G}$, we have $Q_\Pi^\infty(\mathcal{G}) = \bigcup_{i=0}^\infty \varphi_i(\mathcal{G})$. The φ_i's are called the *expansions* of Q. In [25], expansions of a Datalog program Π are described in terms of so-called *expansion trees*, which are finite trees in which each node is labeled with an instance of a rule of Π. We call the head and the body of a node respectively the head and the body of the rule labeling the node. In an expansion tree for an IDB predicate Q, the root is labeled by a rule whose head is a Q-atom. If a node g is labeled by a rule instance

$$R(\mathbf{t}) \leftarrow R_1(\mathbf{t}^1), \ldots, R_m(\mathbf{t}^m)$$

where the IDB atoms in the body of the rule are $R_{i_1}(\mathbf{t}^{i_1}), \ldots, R_{i_\ell}(\mathbf{t}^{i_\ell})$, then g has children $g1, \ldots, g\ell$ labeled with rule instances whose heads are respectively the atoms $R_{i_1}(\mathbf{t}^{i_1}), \ldots, R_{i_\ell}(\mathbf{t}^{i_\ell})$. In particular, if all atoms in the body of g are EDB atoms, then g must be a leaf. The query corresponding to an expansion tree is the conjunction of all EDB atoms in the nodes of the tree, with the variables in the head of the root as the distinguished variables. Thus, we can view an expansion tree τ as a conjunctive query, and extend, in the obvious way, the notion of containment mapping also to mappings from a conjunctive query to an expansion tree. Let $trees(Q, \Pi)$ denote the set of expansion trees for an IDB predicate Q in Π. (Note that $trees(Q, \Pi)$ is, in general, an infinite set.) Then for every database $\mathcal{G}$, we have

$$Q_\Pi^\infty(\mathcal{G}) = \bigcup_{\tau \in trees(Q,\Pi)} \tau(\mathcal{G})$$

It follows that Π is contained in a conjunctive query φ if there is a containment mapping from φ to each expansion tree τ in $trees(Q, \Pi)$.

Unfortunately, the number of variables, and hence the number of node labels in expansion trees is not bounded, and thus expansion trees are not directly suited for an automata-theoretic approach to containment. In [25], the notion of *proof tree* is introduced, with the idea of describing expansion trees using a finite number of labels. The number of labels is bound by bounding the set of variables that can occur in labels of nodes in the tree. If r is a rule of a Datalog program Π, then let $num\_var(r)$ be the number of variables occurring in IDB atoms in r (head or body). Let $num\_var(\Pi)$ be twice the maximum of $num\_var(r)$ for all rules r in Π. Let $var(\Pi)$ be the set $\{x_1, \ldots, x_{num\_var(\Pi)}\}$. A *proof tree* for Π is simply an expansion tree for Π all of whose variables are from $var(\Pi)$. We denote the set of proof trees for a predicate Q of a Datalog program Π by $p\_trees(Q, \Pi)$.

A proof tree represents an expansion tree where variables are re-used. In other words, the same variable is used to represent a set of distinct variables in the expansion tree. Intuitively, to reconstruct an expansion tree for a given proof tree, we need to distinguish among occurrences of variables. Let g_1 and g_2 be nodes in a proof tree τ, with a lowest common ancestor g_0, and let x_1 and x_2 be occurrences, in g_1 and g_2, respectively, of a variable x. We say that x_1 and x_2 are *connected* in τ if the head of every node, except perhaps for g_0, on the simple path connecting g_1 and g_2 has an occurrence of x. (Notice that this means that x also occurs in the body of g_0.) We say that an occurrence x of a variable x in τ is a *distinguished occurrence* if it is connected to an occurrence of x in the head of the root of τ.

We want to define containment mappings from conjunctive queries to proof trees such that there is a containment mapping from a conjunctive query to a proof tree if and only if there is a containment mapping from the conjunctive query to the expansion corresponding to the proof tree. To do so, we need to force a variable in the conjunctive query to map to a unique variable in the expansion corresponding to the proof tree. A *strong containment mapping* from a conjunctive query φ to a proof tree τ is a containment mapping h from φ to τ with the following properties:

- h maps distinguished occurrences in φ to distinguished occurrences in τ, and
- if x_1 and x_2 are two occurrences of a variable x in φ, then the occurrences $h(\mathsf{x}_1)$ and $h(\mathsf{x}_2)$ in τ are connected.

The following characterization of containment of a union of conjunctive queries in a Datalog program was shown in [25].

Theorem 2 ([25]) *Let Π be a Datalog program with goal predicate Q, and let $\Phi = \cup_i \varphi_i$ be a (possibly infinite) union of conjunctive queries over EDB predicates. Then Π is contained in Φ if and only if for every proof tree $\tau \in p\_trees(Q, \Pi)$ there is a strong containment mapping from some φ_i to τ.*

The above theorem is shown in [25] for *finite* unions of conjunctive queries only. However, it is easy to see that the proof carries through also for infinite unions.

Notice that, together with Theorem 1, Theorem 2 by itself does not provide decidability of containment of Datalog in (possibly infinite) unions of conjunctive queries, since one needs a method to check the existence of a strong containment mapping. Undecidability of containment between Datalog queries [22] shows that such a method will not exist in general for (infinite) unions that are expansions of Datalog programs. However, in [25] the above result is exploited to show that containment of a Datalog query in a finite union of conjunctive queries is in 2EXPTIME (and in fact 2EXPTIME-complete).

To exploit Theorem 2 for containment of Datalog queries in union of C2RPQs, we need to characterize the problem in terms of containment between Datalog and (infinite) unions of conjunctive queries. An *expansion* of a C2RPQ

$$Q(x_1, \ldots, x_n) \leftarrow E_1(y_1, y_1'), \ldots, E_m(y_m, y_m')$$

is a CQ of the form

$$\begin{aligned} Q(x_1, \ldots, x_n) \leftarrow\ & r_1^1(y_1, z_1^1), r_1^2(z_1^1, z_1^2), \ldots, r_1^{n_1}(z_1^{n_1-1}, y_1'), \\ & \vdots \\ & r_m^1(y_m, z_m^1), r_m^2(z_m^1, z_m^2), \ldots, r_m^{n_m}(z_m^{n_m-1}, y_m') \end{aligned}$$

where, for each $i \in \{1, \ldots, m\}$, we have that $n_i \geq 0$, that $r_i^1 \cdots r_i^{n_i} \in L(E_i)$, and that all variables z_i^j are pairwise distinct. Observe that, when $n_i = 0$, we have that $r_i^1 \cdots r_i^{n_i} = \varepsilon$, and $r_i^1(y_i, z_i^1), r_i^2(z_i^1, z_i^2), \ldots, r_i^{n_i}(z_i^{n_i-1}, y_i')$ becomes simply $y_i = y_i'$. Notice that, a C2RPQ has in general many expansions, and that, due to transitive closure, the number of such expansions may be infinite.

The following lemma is an easy consequence of Theorem 2 and of the semantics of unions of C2RPQs.

Lemma 3 *Let Π be a Datalog program with goal predicate Q, and let $\Gamma = \cup_i \gamma_i$ be a finite union of C2RPQs. Then Π is contained in Γ if and only if for every proof tree $\tau \in p\_trees(Q, \Pi)$ there is a C2RPQ γ_i of Γ and an expansion φ of γ_i such that there is a strong containment mapping from φ to τ.*

In the following, we show how to check this condition using tree automata. Unlike [25], where standard (one-way) nondeterministic tree automata are adopted, we need to resort to two-way alternating tree automata. This is due to the presence of inverses of relations in 2RPQs, and due to the fact that in 2RPQs (and hence in C2RPQs) concatenation and transitive closure introduce implicit variables that need to be dealt with.

4 Two-way Alternating Tree Automata

We present the basic notions on automata used in the rest of the paper. We assume familiarity with the standard notions of (one-way) word automata (1NFAs) [38] and (one-way) nondeterministic tree automata (1NTAs) [39], and concentrate on two-way alternating tree automata (2ATAs).

Trees are represented as prefix closed finite sets of words over $\mathbb{N}_+$ (the set of

positive natural numbers). Formally, a *tree* T is a finite subset of $\mathbb{N}_+$, such that if $g{\cdot}c \in T$, where $g \in \mathbb{N}_+^*$ and $c \in \mathbb{N}_+$, then also $g \in T$ and if $c > 1$ then also $g{\cdot}(c-1) \in T$. The elements of T are called *nodes*, and for every $g \in T$, the nodes $g{\cdot}c \in T$, with $c \in \mathbb{N}_+$, are the *successors* of g. By convention we take $g{\cdot}0 = g$, and $g{\cdot}c{\cdot}(-1) = g$. By definition, the empty sequence ε is a member of every tree, and is called the *root*. Note that $\varepsilon \cdot -1$ is undefined. The *branching degree* $d(g)$ of a node g denotes the number of successors of g. If the branching degree of all nodes of a tree is bounded by k, we say that the tree has branching degree k. Given a finite alphabet Σ, a Σ-*labeled tree* τ is a pair (T, V), where T is a tree and $V : T \to \Sigma$ maps each node of T to an element of Σ. Σ-labeled trees are often referred to as *trees*, and if $\tau = (T, V)$ is a (labeled) tree and g is a node of T, we use $\tau(g)$ to denote $V(g)$.

Two-way alternating tree automata (2ATAs) [35,23], are a generalization of standard nondeterministic top-down tree automata (1NTAs) [40,41]) with both upward moves and with alternation. Let $\mathcal{B}(I)$ be the set of positive Boolean formulae over I, built inductively by applying $\wedge$ and $\vee$ starting from **true**, **false**, and elements of I. For a set $J \subseteq I$ and a formula $\varphi \in \mathcal{B}(I)$, we say that J *satisfies* φ if and only if, assigning **true** to the elements in J and **false** to those in $I \setminus J$, makes φ true. For a positive integer k, let $[k] = \{-1, 0, 1, \ldots, k\}$. A *two-way alternating tree automaton* (2ATA) over a finite alphabet Σ running over trees with branching degree k, is a tuple $\mathbf{A} = (\Sigma, S, \delta, s_0, F)$, where S is a finite set of states, $\delta : S \times \Sigma \to \mathcal{B}([k] \times S)$ is the transition function, $s_0 \in S$ is the initial state, and $F \subseteq S$ is the set of final states. The transition function maps a state $s \in S$ and an input letter $\sigma \in \Sigma$ to a positive Boolean formula φ over $[k] \times S$. Since φ can be written in conjunctive normal form, in the following we view it as a set of conjunctions. Intuitively, when the 2ATA performs a transition, it nondeterministically chooses one of the conjunctions in φ, and then, for each pair (c, s') appearing in φ a new copy of the automaton starts in state s' and moves to the direction suggested by c.

A run ν of a 2ATA $\mathbf{A}$ over a labeled tree $\tau = (T, V)$ is a labeled tree (T_ν, V_ν) in which every node is labeled by an element of $T \times S$. A node f of T_ν labeled by (g, s) describes a copy of $\mathbf{A}$ that is in the state s and reads the node g of τ. The labels of adjacent nodes have to satisfy the transition function of $\mathbf{A}$. Formally, a run (T_ν, V_ν) is a $(T \times S)$-labeled tree satisfying:

(1) $\varepsilon \in T_\nu$ and $V_\nu(\varepsilon) = (\varepsilon, s_0)$.
(2) Let $f \in T_\nu$, with $V_\nu(f) = (g, s)$ and $\delta(s, V(g)) = \varphi$. Then there is a (possibly empty) set $C = \{(c_1, s_1), \ldots, (c_n, s_n)\} \subseteq [k] \times S$ such that:
   - C satisfies φ and
   - for all $i \in \{1, \ldots, n\}$, we have that $f{\cdot}i \in T_\nu$, $g{\cdot}c_i$ is defined, and $V_\nu(f{\cdot}i) = (g{\cdot}c_i, s_i)$.

A run $\nu = (T_\nu, V_n u)$ on a tree τ is *accepting* if, whenever a leaf of T_ν is labeled

by (g, s), then $s \in F$. $\mathbf{A}$ *accepts* a labeled tree τ if it has an accepting run on τ. The set of trees accepted by $\mathbf{A}$ is denoted $\mathcal{T}(\mathbf{A})$. The *nonemptiness* problem for tree automata consists in deciding, given a tree automaton $\mathbf{A}$, whether $\mathcal{T}(\mathbf{A})$ is nonempty.

As shown in [23], 2ATAs can be converted to complementary 1NTAs with only a single exponential blowup. Moreover, it is straightforward to see that one can construct a 2ATA of polynomial size accepting the finite union of the languages accepted by n 2ATAs.

Proposition 4 ([23]) *Given a 2ATA $\mathbf{A}$ over an alphabet Σ, there is a 1NTA $\overline{\mathbf{A}}$ of size exponential in the size of $\mathbf{A}$ such that $\overline{\mathbf{A}}$ accepts a Σ-labeled tree τ if and only if τ is rejected by $\mathbf{A}$.*

Proposition 5 *Given n 2ATAs $\mathbf{A}_1, \ldots, \mathbf{A}_n$ over an alphabet Σ, there is a 2ATA $\mathbf{A}_\cup$ of size linear in the sum of the sizes of $\mathbf{A}_1, \ldots, \mathbf{A}_n$ such that $\mathcal{T}(\mathbf{A}_\cup) = \mathcal{T}(\mathbf{A}_1) \cup \cdots \cup \mathcal{T}(\mathbf{A}_n)$.*

We make also use of the following standard results for 1NTAs.

Proposition 6 ([42]) *Given 1NTAs $\mathbf{A}_1$ and $\mathbf{A}_2$ over an alphabet Σ, there is a 1NTA $\mathbf{A}_\cap$ whose size is the product of the sizes of $\mathbf{A}_1$ and $\mathbf{A}_2$ such that $\mathcal{T}(\mathbf{A}_\cap) = \mathcal{T}(\mathbf{A}_1) \cap \mathcal{T}(\mathbf{A}_2)$.*

Proposition 7 ([40,41]) *The nonemptiness problem for 1NTAs is decidable in polynomial time.*

In fact, the nonemptiness problem for 1NTAs is decidable in linear time, see [43,44]

5 Containment of Datalog in Unions of C2RPQs

The main feature of proof trees is the fact that the number of possible labels is finite; it is actually exponential in the size of Π. Because the set of labels is finite, the set of proof trees $p\_trees(Q, \Pi)$, for an IDB predicate Q in a program Π, can be described by a tree automaton.

Theorem 8 ([25]) *Let Π be a Datalog program with a goal predicate Q. Then there is a 1NTA $\mathbf{A}_{Q,\Pi}^{p\_trees}$, whose size is exponential in the size of Π, such that $\mathcal{T}(\mathbf{A}_{Q,\Pi}^{p\_trees}) = p\_trees(Q, \Pi)$.*

The automaton $\mathbf{A}_{Q,\Pi}^{p\_trees} = (\Sigma, \mathcal{I} \cup \{accept\}, \mathcal{I}_Q, \delta, \{accept\})$, analogous to the one defined in [25], is as follows.

The state set is the set $\mathcal{I}$ of all IDB atoms with variables among $var(\Pi)$, plus an accepting state. The start-state set $\mathcal{I}_Q$ is the set of all atoms $Q(\mathbf{s})$, where the variables of $\mathbf{s}$ are in $var(\Pi)$. The alphabet Σ is is the set $\mathcal{R}$ of instances of rules of Π over $var(\Pi)$. The transition function δ is constructed as follows. Let ϱ be the body of a rule instance in $\mathcal{R}$

$$R(\mathbf{t}) \ \leftarrow \ R_1(\mathbf{t}^1), \ldots, R_m(\mathbf{t}^m)$$

- If the IDB atoms in ϱ are $R_{i_1}(\mathbf{t}^{i_1}), \ldots, R_{i_\ell}(\mathbf{t}^{i_\ell})$, then there is a transition [3]

$$\langle 1, R_{i_1}(\mathbf{t}^{i_1})\rangle \wedge \cdots \wedge \langle \ell, R_{i_\ell}(\mathbf{t}^{i_\ell})\rangle \ \in \ \delta(R(\mathbf{t}), (R(\mathbf{t}) \leftarrow \varrho))$$

- If all atoms in ϱ are EDB atoms, then there is a transition

$$\langle 0, accept\rangle \ \in \ \delta(R(\mathbf{t}), (R(\mathbf{t}) \leftarrow \varrho))$$

It is easy to see that the number of states and transitions in $\mathbf{A}^{p\_trees}_{Q,\Pi}$ is exponential in the size of Π.

We now show that strong containment of proof trees in a C2RPQ can be checked by tree automata as well. Let Π be a Datalog program with (binary) EDB predicates in Δ and with goal predicate Q, and let γ be a C2RPQ over $\Delta^{\pm}$ of the same arity as Q. We describe the construction of a 2ATA $\mathbf{A}^{\gamma}_{Q,\Pi}$ that accepts all proof trees τ in $p\_trees(Q,\Pi)$ such that there is an expansion φ of γ and a strong containment mapping from φ to τ.

We view γ as a set of atoms $E(x,y)$, where E is a 1NFA $E = (\Delta^{\pm}, S_E, s_E, \delta_E, F_E)$, with $s_E \in S_E$ and $F_E \subseteq S_E$, and where, w.l.o.g., δ_E does not contain ε-transitions. Also, w.l.o.g., we assume that for two distinct atoms $E_1(x_1,y_1)$ and $E_2(x_2,y_2)$, E_1 and E_2 are distinct automata with disjoint sets of states, i.e., $S_{E_1} \cap S_{E_2} = \emptyset$. For a 1NFA E, we use E^F_s to denote the 1NFA identical to E, except that $s \in S_E$ is the initial state of E^F_s, and $F \subseteq S_E$ is the set of final states of E^F_s. When F is a singleton, we may omit set brackets.

Let V_γ be the set of variables appearing in the C2RPQ γ, and $V^+_\gamma = \{\bar{v}^1_E, \bar{v}^2_E \mid E(x,y) \in \gamma\}$, i.e., for each 1NFA $E(x,y) \in \gamma$, V^+_γ contains two special variables $\bar{v}^1_E$ and $\bar{v}^2_E$. We denote with $\mathcal{B}$ the collection of all sets β of atoms, such that β contains, for each atom $E(x,y) \in \gamma$, at most one atom $E^F_s(x',y')$, for some $s \in S_E$ and $F \subseteq S_E$, with x' either x or $\bar{v}^1_E$ and y' either y or $\bar{v}^2_E$. Notice that the size of $\mathcal{B}$ is exponential in the size of γ. Indeed, let k be the number of atoms in γ and let m be an upper bound on the number of states of each 1NFA in γ. All possible variants of a 1NFA obtained by changing the initial state

[3] For uniformity, we use the notation of 2ATAs to denote the transitions of 1NTAs.

and/or final states are $m \cdot 2^m$. Hence, the number of possible sets of 1NFAs of at most k elements is $(m \cdot 2^m)^k = 2^{O(m \cdot k)}$.

The automaton $\mathbf{A}^{\gamma}_{Q,\Pi}$ is $(\Sigma, S \cup \{accept\}, S_Q, \delta, \{accept\})$.

- The alphabet Σ is again the set $\mathcal{R}$ of instances of rules of Π over $var(\Pi)$.
- The state set S is the set $\mathcal{I} \times \mathcal{B} \times 2^{V_\gamma \times var(\Pi)} \times 2^{V^+_\gamma \times var(\Pi)}$. Recall that $\mathcal{I}$ is the set of all IDB atoms with variables among $var(\Pi)$. The second component represents the collection of automata accepting sequences of atoms that have to be mapped to atoms in the tree τ accepted by $\mathbf{A}^{\gamma}_{Q,\Pi}$, and the third and fourth components contain the set of partial mappings respectively from V_γ and V^+_γ to $var(\Pi)$.
- The start-state set S_Q consists of all tuples $(Q(\mathbf{s}), \gamma, M_{\gamma,\mathbf{s}}, \emptyset)$, where the variables of $\mathbf{s}$ are in $var(\Pi)$ and $M_{\gamma,\mathbf{s}}$ is a mapping of the distinguished variables of γ into the variables of $\mathbf{s}$.

The transition function δ of $\mathbf{A}^{\gamma}_{Q,\Pi}$ is constructed as follows. Let ϱ be the body of a rule instance in $\mathcal{R}$

$$R(\mathbf{t}) \leftarrow R_1(\mathbf{t}^1), \ldots, R_m(\mathbf{t}^m)$$

(1) There is an "atom mapping" transition

$$\langle 0, (R(\mathbf{t}), \beta', M, M'_+)\rangle \ \in \ \delta((R(\mathbf{t}), \beta, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

if there is an EDB atom $e(a,b)$ among $R_1(\mathbf{t}^1), \ldots, R_m(\mathbf{t}^m)$ and if β' coincides with β, except that one element $E^F_s(x,y)$ in β is replaced in β' by $E^F_{s'}(x',y)$, and one of the following holds:

- $s' \in \delta_E(s,e)$ and
  - if $x \in V_\gamma$ (i.e., x is a variable of γ), M maps x to a, and M_+ does not map $\bar{v}^1_E$, then $x' = \bar{v}^1_E$ and $M'_+ = M_+ \cup \{(\bar{v}^1_E, b)\}$;
  - if $x = \bar{v}^1_E \in V^+_\gamma$ (i.e., x is the first special variable for the 1NFA E) and $(\bar{v}^1_E, a) \in M_+$, then $x' = x = \bar{v}^1_E$, and $M'_+ = M_+ \setminus \{(\bar{v}^1_E, a)\} \cup \{(\bar{v}^1_E, b)\}$;
- $s' \in \delta_E(I, e^-)$ and
  - if $x \in V_\gamma$ (i.e., x is a variable of γ), M maps x to b, and M_+ does not map $\bar{v}^1_E$, then $x' = \bar{v}^1_E$ and $M'_+ = M_+ \cup \{(\bar{v}^1_E, a)\}$;
  - if $x = \bar{v}^1_E \in V^+_\gamma$ (i.e., x is the first special variable for the 1NFA E) and $(\bar{v}^1_E, b) \in M_+$, then $x' = x = \bar{v}^1_E$, and $M'_+ = M_+ \setminus \{(\bar{v}^1_E, b)\} \cup \{(\bar{v}^1_E, a)\}$.

Intuitively, an "atom mapping" transition maps the next atom read by some 1NFA in β to some EDB atom in ρ, and modifies M_+ accordingly. Note that the variable x (either a variable of V_γ or the special variable $\bar{v}^1_E$) must already be mapped (respectively by M or M_+) to some variable in the current node of τ.

(2) There is a "splitting" transition

$$\langle 0, (R(\mathbf{t}), \beta', M, M'_+)\rangle \ \wedge \ \langle 0, (R(\mathbf{t}), \beta'', M, M''_+)\rangle \ \in \\ \delta((R(\mathbf{t}), \beta, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

if the following hold:

- M'_+ and M''_+ coincide with M_+, except for the changes described in the following point;
- β can be partitioned into β_1, β_2, and β_3; moreover $\beta' = \beta_1 \cup \beta'_3$ and $\beta'' = \beta_2 \cup \beta''_3$, where β'_3 and β''_3 are sets of elements that consist of one element for each element $E^F_s(x, y)$ in β_3, obtained as follows: for some state s' of E and some variable $a \in var(\Pi)$ appearing in $R(\mathbf{t}) \leftarrow \varrho$, one of the following holds:
  - β'_3 contains the element $E^{s'}_s(x, \bar{v}^2_E)$, β''_3 contains the element $E^F_{s'}(\bar{v}^1_E, y)$, M'_+ (re-)maps $\bar{v}^2_E$ to a, and M''_+ (re-)maps $\bar{v}^1_E$ to a;
  - β'_3 contains the element $E^F_{s'}(\bar{v}^1_E, y)$, β''_3 contains the element $E^{s'}_s(x, \bar{v}^2_E)$, M'_+ (re-)maps $\bar{v}^1_E$ to a, and M''_+ (re-)maps $\bar{v}^2_E$ to a;
- β' and β'' can share a variable in V_γ only if this variable is in the domain of M. (Notice that two occurrences of a special variable in V^+_γ shared by β' and β'' are not related to each other.)

A "splitting" transition partitions the atoms in β into two parts. The goal is to enable the two parts to be manipulated separately. For example, one part may correspond to those atoms that are intended to be "moved" together to an adjacent node in a future transition, while the other part may correspond to those atoms that are meant to stay together in the current node for further processing, e.g., by further splitting or by mapping to EDB atoms. During splitting, some atoms in β may be actually split into two subatoms. The mappings M and M_+ have to "bind" together variables that are in common to the two conjuncts of the transition.

(3) There is a "downward moving" transition

$$\langle j, (R_{i_j}(\mathbf{t}^{i_j}), \beta, M, M_+)\rangle \ \in \ \delta((R(\mathbf{t}), \beta, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

if $j \in \{1, \ldots, \ell\}$, where ℓ is the number of IDB atoms in ϱ and $R_{i_j}(\mathbf{t}^{i_j})$ is the j-th IDB atom, and if for all variables that occur in β and that are in the domain of either M or M_+, their image is in $\mathbf{t}^{i_j}$.

A "downward moving" transition moves to a successor node, and is intended to be applied whenever no next atom can be mapped and no further splitting is possible. Moving is possible only if variables that are both in atoms still to be mapped (and thus in β) and have already been mapped (and thus are in the domain of either M or M_+) can be propagated through the head of the rule to which the automaton moves.

(4) There is an "upward moving" transition

$$\langle -1, (R'(\mathbf{t}'), \beta, M, M_+)\rangle \ \in \ \delta((R(\mathbf{t}), \beta, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

if $R'(\mathbf{t}')$ is the head of some rule instance and if for all variables that occur in β and that are in the domain of either M or M_+, their image is in $\mathbf{t}$.

An "upward moving" transition is similar to a "downward moving" one, except that it moves to the predecessor node. Notice that, after an "upward moving" transition, the automaton will be able to perform further moves (and hence to eventually accept) only if the head of the rule instance in the predecessor node is $R'(\mathbf{t}')$.

(5) There is an "equality checking" transition

$$\langle 0, (R(\mathbf{t}), \beta', M, M_+)\rangle \ \in \ \delta((R(\mathbf{t}), \beta, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

if the following hold:

- β can be partitioned into β_0 and β';
- for all atoms $E_s^F(x, y) \in \beta_0$ we have that
  - $s \in F$,
  - (x, a) and (y, a) are in $M \cup M_+$, for some variable a in ϱ or $\mathbf{t}$, i.e., both x and y are in the domain of M or of M_+ and they are mapped to the same variable a;

An "equality checking" transition gets rid of those elements in β all of whose atoms have already been mapped to atoms in τ. While doing so, it checks that M and M_+ are compatible with the equalities induced by such atoms.

(6) There is a "mapping extending" transition

$$\langle 0, (R(\mathbf{t}), \beta, M', M_+)\rangle \ \in \ \delta((R(\mathbf{t}), \beta, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

if M' is a partial mapping that extends M.

A "mapping extending" transition adds some variables to the mapping M. This may be necessary to be able to apply some other transition that requires certain variables to appear in M.

(7) There is a "final" transition

$$\langle 0, \mathit{accept}\rangle \ \in \ \delta((R(\mathbf{t}), \emptyset, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

A "final" transition moves to the accepting state whenever there are no further atoms in β that have to be processed.

It is easy to see that the number of states and transitions in $\mathbf{A}^{\gamma}_{Q,\Pi}$ is exponential in the size of Π and γ. The following two basic lemmas establish the correctness of the above construction.

Lemma 9 *Let τ be a proof tree in p_trees(Q, Π). If there is an expansion φ of γ and a strong containment mapping h from φ to τ, then τ is accepted by $\mathbf{A}^{\gamma}_{Q,\Pi}$.*

Proof. We prove acceptance by showing the existence of an accepting run ν of $\mathbf{A}^{\gamma}_{Q,\Pi}$. We view the expansion φ of γ as a set of sequences of atoms; for each atom $E(x,y)$ in the body of γ, φ contains one sequence of atoms

$$\varphi_E \;=\; r^1(z^0,z^1),\, r^2(z^1,z^2),\, \ldots,\, r^n(z^{n-1},z^n)$$

with $z^0 = x$, $z^n = y$, and $r^1 \cdots r^n \in L(E)$.

Besides the accepting run ν we make use of a tree W, with the same set of nodes as ν, in which each node is labeled by a set of sequences of atoms of φ. More precisely, for each node f, each $w \in W(f)$ is a (possibly empty) subsequence of some sequence φ_E in φ, representing those atoms in φ_E that, when $\mathbf{A}^{\gamma}_{Q,\Pi}$ is at node f of the run, have not already been mapped to atoms in τ. For the root ε we have that $W(\varepsilon) = \varphi$.

Let f be a node of the run ν with $\nu(f) = (g, (R(\mathbf{t}), \beta, M, M_+))$, where g is a node of τ and $\tau(g) = (R(\mathbf{t}) \leftarrow \varrho)$. We say that $W(f)$ is *compatible* with β if it consists of sequences of atoms, one sequence $w = r^u(z^{u-1},z^u), r^{u+1}(z^u,z^{u+1}), \ldots, r^v(z^{v-1},z^v)$ for each atom $E^F_s(x',y')$ in β, where w is a contiguous subsequence of the sequence $\varphi_E = r^1(z^0,z^1), r^2(z^1,z^2), \ldots, r^n(z^{n-1},z^n)$ in φ corresponding to the atom $E(x,y)$ in γ, and we have that

- $r^u \cdots r^v \in L(E^F_s)$;
- $u > 1$ iff $x' = \bar{v}^1_E$, and if $u > 1$ then $(\bar{v}^1_E, h(z^{u-1})) \in M_+$;
- $v < n$ iff $y' = \bar{v}^2_E$, and if $v < n$ then $(\bar{v}^2_E, h(z^v)) \in M_+$;

For each atom $E^F_s(x,y)$ in β for which the corresponding sequence of atoms in $W(f)$ is $r^u(z^{u-1},z^u), \ldots, r^v(z^{v-1},z^v)$, we use $\vartheta(x)$ to denote z^{u-1} and $\vartheta(y)$ to denote z^v. Notice that, if $x \in V_\gamma$, then $\vartheta(x)$ denotes x itself. We say that f is *connected*, if for each variable x appearing both in β and in the domain of $M \cup M_+$, $R(\mathbf{t}) \leftarrow \varrho$ contains an occurrence of a variable connected to the occurrence $h(\vartheta(x))$. We say that M_+ is *compatible* with β, if for each $E^F_s(x,y)$ in β, if x is in V_γ, then M_+ does not map $\bar{v}^1_E$; similarly for y. We say that the pair (g,s), with g a node of τ and s a state of $\mathbf{A}^{\gamma}_{Q,\Pi}$, is *accepting* (for $\mathbf{A}^{\gamma}_{Q,\Pi}$ and τ) if there is an accepting run ν of $\mathbf{A}^{\gamma}_{Q,\Pi}$ on τ and a node f of ν such that $\nu(f) = (g,s)$.

We show that $\nu(f) = (g, (R(\mathbf{t}), \beta, M, M_+))$ is accepting for $\mathbf{A}^{\gamma}_{Q,\Pi}$ and τ, if the following conditions hold:

(1) M is consistent with h and maps all distinguished variables of γ;
(2) M_+ is compatible with β;
(3) f is connected;
(4) $W(f)$ is compatible with β.

We proceed by induction on $S(f)$, where $S(f)$ is the sum of the lengths of the sequences of atoms in $W(f)$. We count an equality atom as having length 1, and a sequence consisting of n atoms different from equalities as having length $n+1$. Below we consider the case where conditions (1) to (4) are satisfied for a node f. If they are not, the property we are proving trivially holds.

- Base case: $S(f) = 0$. Then $W(f)$ is empty. So is β, and $\mathbf{A}^{\gamma}_{Q,\Pi}$ can perform a "final" transition to the accepting state. Hence $\nu(f)$ is accepting.
- Inductive case 1: Assume there is a nonempty sequence $w = r^u(z^{u-1}, z^u), r^{u+1}(z^u, z^{u+1}), \ldots, r^v(z^{v-1}, z^v)$ in $W(f)$ with $r^u \cdots r^v \in L(E^F_s)$ for some $E^F_s(x, y)$ in β, such that the body ϱ of the rule of the node g of the proof tree contains an atom $r^u(a, b)$ and we have that $h(z^{u-1}) = a$ and $h(z^u) = b$. In other words, h maps the first atom in w to an atom in ϱ. We consider the case where $r^u = e$, for some $e \in \Delta$ and where $x \in V_\gamma$ (i.e., $u = 1$ and $x = z^0$ is a variable of γ). The other cases can be dealt with analogously.

  Since $r^u \cdots r^v \in L(E^F_s)$, there must be some $s' \in \delta_E(s, e)$ such that $r^{u+1} \cdots r^v \in L(E^F_{s'})$. Since M_+ is compatible with β it does not map $\bar{v}^1_E$.

  Assume that x is in the domain of M. Since M is consistent with h it maps x to a. Then $\mathbf{A}^{\gamma}_{Q,\Pi}$ can perform an "atom mapping" transition

$$\langle 0, (R(\mathbf{t}), \beta', M, M'_+)\rangle \ \in \ \delta((R(\mathbf{t}), \beta, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

  where β' coincides with β, except that the element $E^F_s(x, y)$ in β is replaced in β' by $E^F_{s'}(\bar{v}^1_E, y)$, and $M'_+ = M_+ \cup \{(\bar{v}^1_E, b)\}$.

  Hence, in the run ν there is a unique successor $f{\cdot}1$ of f with $\nu(f{\cdot}1) = (g, (R(\mathbf{t}), \beta', M', M'_+))$. For W we have that $W(f{\cdot}1)$ coincides with $W(f)$, except that the sequence w in $W(f)$ is replaced in $W(f{\cdot}1)$ by $w' = r^{u+1}(z^u, z^{u+1}), \ldots, r^v(z^{v-1}, z^v)$. Observe that $f{\cdot}1$ satisfies conditions (1) to (4) and that $S(f{\cdot}1) = S(f) - 1$, since we have mapped one atom of $W(f)$. Thus, by inductive hypothesis, $\nu(f{\cdot}1)$ is accepting, and hence also $\nu(f)$.

  If x is not in the domain of M, then $\mathbf{A}^{\gamma}_{Q,\Pi}$ can first perform a "mapping extending" transition

$$\langle 0, (R(\mathbf{t}), \beta, M', M_+)\rangle \ \in \ \delta((R(\mathbf{t}), \beta, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

  such that $M' = M \cup \{(x, a)\}$, and then perform the transition above. In this case, M' is still consistent with h and the resulting node in the run is connected, since x is mapped to a variable in $R(\mathbf{t}) \leftarrow \varrho$.
- Inductive case 2: Assume there is a sequence $w = r^u(z^{u-1}, z^u), \ldots, r^v(z^{v-1}, z^v)$ in $W(f)$ that collapses to the equality $z^{u-1} = z^v$, being $r^u \cdots r^v = \varepsilon$, and such that h maps z^{u-1} and z^v to occurrences of a variable both connected to the same variable a in $R(\mathbf{t}) \leftarrow \varrho$.

  We consider only the case where z^{u-1} is a variable x in V_γ and and z^v is

a variable y in V_γ. The other cases are analogous. For each $E_s^F(x,y)$ in β corresponding to w, we have that $s \in F$.

Assume that both x and y are in the domain of M. Since M is consistent with h, it maps both x and y to a. Thus $\mathbf{A}_{Q,\Pi}^{\gamma}$ can perform the "equality checking" transition

$$\langle 0, (R(\mathbf{t}), \beta', M, M_+)\rangle \ \in \ \delta((R(\mathbf{t}), \beta, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

Hence, in the run ν there is a unique successor $f{\cdot}1$ of f with $\nu(f{\cdot}1) = (g, (R(\mathbf{t}), \beta', M, M_+))$. For W we have that $W(f{\cdot}1)$ coincides with $W(f) \setminus \{w\}$. Observe that $f{\cdot}1$ satisfies conditions (1) to (4) and that $S(f{\cdot}1) = S(f) - 1$. Thus, by inductive hypothesis, $\nu(f{\cdot}1)$ is accepting, and hence also $\nu(f)$.

If x or y is not in the domain of M, then $\mathbf{A}_{Q,\Pi}^{\gamma}$ can first perform a "mapping extending" transition, as in the previous case.

- Inductive case 3: Consider some $j \in \{-1, 1, \ldots, \ell\}$, where ℓ is the number of IDB atoms in ϱ. Let W_1 be the subset of $W(f)$ consisting of those sequences w of atoms such that h maps both the first and the last atom of w to atoms in the j-th subtree of g, where we take the -1-th subtree of g to be τ without the tree rooted at g. Let W_2 be the subset of $W(f)$ consisting of those sequences w of atoms such that h maps neither the first nor the last atom of w to atoms in the j-th subtree of g. Let W_3 be the remaining sequences of atoms in $W(f)$. Finally, let both W_1 and W_2 be different from $W(f)$.

  We have a corresponding partition of β into β_1, β_2, and β_3. For each sequence $w = r^u(z^{u-1}, z^u), \ldots, r^v(z^{v-1}, z^v)$ in W_3 corresponding to an element $E_s^F(x,y)$ in β_3, since $r^u(z^{u-1}, z^u)$ is mapped to an atom in the j-th subtree of g, and $r^v(z^{v-1}, z^v)$ is mapped elsewhere, there must be some intermediate variable z^i in the sequence that is mapped by h to an occurrence of a variable connected to a variable a in $R(\mathbf{t}) \leftarrow \varrho$. (The case where the last atom is mapped to the j-subtree is analogous.) Hence there must be a state s' of E such that $r^u \cdots r^i \in L(E_s^{s'})$ and $r^{i+1} \cdots r^v \in L(E_{s'}^F)$.

  Let β_3' be obtained from β_3 by replacing each atom $E_s^F(x,y)$ with one of $E_s^{s'}(x, \bar{v}_2)$ or $E_{s'}^F(\bar{v}_E^1, y)$, depending on whether the first or the last atom of the corresponding sequence in W_3 is mapped to the j-th subtree. Let β_3'' be defined the other way round. Finally, let $\beta' = \beta_1 \cup \beta_3'$ and $\beta'' = \beta_2 \cup \beta_3''$.

  Assume that all variables of V_γ shared by β' and β'' are already in the domain of M. Thus $\mathbf{A}_{Q,\Pi}^{\gamma}$ can perform the "splitting" transition

$$\langle 0, (R(\mathbf{t}), \beta', M, M_+')\rangle \ \wedge \ \langle 0, (R(\mathbf{t}), \beta'', M, M_+'')\rangle \ \in \ \delta((R(\mathbf{t}), \beta, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

  with M_+' and M_+'' defined as required.

  Hence, in the run ν there are two successors $f{\cdot}1$ and $f{\cdot}2$ of f with $\nu(f{\cdot}1) = (g, (R(\mathbf{t}), \beta', M, M_+'))$ and $\nu(f{\cdot}1) = (g, (R(\mathbf{t}), \beta'', M, M_+''))$. For

W we have that $W(f{\cdot}1)$ consists of W_1 union the set of subsequences of W_3 corresponding to the elements in β'_3. Analogously for $W(f{\cdot}2)$.

Observe that $f{\cdot}1$ and $f{\cdot}2$ satisfy conditions (1) to (4) above, and that both $S(f{\cdot}1)$ and $S(f{\cdot}2)$ are strictly smaller than $S(f)$, since W_1 and W_2 are by assumption both different from $W(f)$. Thus, by inductive hypothesis, $\nu(f{\cdot}1)$ and $\nu(f{\cdot}2)$ are both accepting, and hence also $\nu(f)$.

If some variable x in V_γ is shared by β' and β'' but is not already in the domain of M, then $\mathbf{A}^\gamma_{Q,\Pi}$ can first perform a "mapping extending" transition, by adding $(x, h(x))$ to M. Observe that, since x in is shared by β' and β'', one occurrence of x must be mapped by h in the j-th subtree, and one somewhere else. Hence, since h is a strong containment mapping, the two occurrences of $h(x)$ must be connected, and so also $R(\mathbf{t}) \leftarrow \varrho$ must contain a connected occurrence of $h(x)$. Thus the node in the run resulting from the "mapping extending" transition is also connected.

- Inductive case 4: When the conditions for the application of the base case and the inductive cases 1 to 3 do not hold, then $W(f)$ is still not empty but we cannot progress with the mapping on the current node g. Since none of the above cases apply it must be that for all variables x in β, $\vartheta(x)$ is mapped by h to a variable in the j-th subtree of g, for some j. Since f is connected, all variables that appear both in β and in the domain of $M \cup M_+$ must be mapped by h to occurrence of variables connected to a variable in $R(\mathbf{t}) \leftarrow \varrho$. Since h is a strong containment mapping and all these variables are mapped in the j-th subtree, if follows that they are connected through variables of the j-th IDB atom in ϱ (respectively, the head $R(\mathbf{t})$, if $j = -1$).

  Thus $\mathbf{A}^\gamma_{Q,\Pi}$ can perform either a "downward moving" transition

  $$\langle j, (R_{i_j}(\mathbf{t}^{i_j}), \beta, M, M_+)\rangle \ \in \ \delta((R(\mathbf{t}), \beta, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

  or an "upward moving" transition

  $$\langle -1, (R'(\mathbf{t}'), \beta, M, M_+)\rangle \ \in \ \delta((R(\mathbf{t}), \beta, M, M_+), (R(\mathbf{t}) \leftarrow \varrho))$$

  Hence, in the run ν there is a unique successor $f{\cdot}1$ of f with $\nu(f{\cdot}1) = (g{\cdot}j, (R_{i_j}(\mathbf{t}^{i_j}), \beta, M, M_+))$ (resp., $\nu(f{\cdot}1) = (g{\cdot}(-1), (R'(\mathbf{t}'), \beta, M, M_+))$). For W we have that $W(f{\cdot}1)$ coincides with $W(f)$. $f{\cdot}1$ satisfies conditions (1) to (4). Moreover, it is easy to see that one can perform transitions only a finite number of times since it is not possible that h requires to pass twice through the same node g of τ. After such transitions, one of the cases above applies. Hence, by inductive hypothesis, $\nu(f)$ is accepting.

Finally, we observe that conditions (1), to (4) are trivially satisfied at the root ε of ν and W. The claim follows. □

Lemma 10 *Let τ be a proof tree in p_trees(Q, Π). If τ is accepted by $\mathbf{A}^\gamma_{Q,\Pi}$, then there is an expansion φ of γ and a strong containment mapping from φ to τ.*

Proof. We construct from an accepting run ν of $\mathbf{A}^{\gamma}_{Q,\Pi}$ an expansion φ and a strong containment mapping from φ to τ. We proceed by bottom-up induction on the run, making use of a tree W analogous to the one used in the proof of Lemma 9, and a tree h with the same set of nodes as ν and W, and such that $h(f)$ is a strong containment mapping from the atoms in $W(f)$ to atoms in τ. In the following, let f be a node of ν with $\nu(f) = (g, (R(\mathbf{t}), \beta, M, M_+))$ and $\tau(g) = (R(\mathbf{t}) \leftarrow \varrho)$.

We recall the definition of $\vartheta(x)$, of connected node of a run, and of accepting pair (g, s). For each atom $E^F_s(x, y)$ in β for which the corresponding sequence of atoms in $W(f)$ is $r^u(z^{u-1}, z^u), \ldots, r^v(z^{v-1}, z^v)$, we use $\vartheta(x)$ to denote z^{u-1} and $\vartheta(y)$ to denote z^v. We say that f is *connected*, if for each variable x appearing both in β and in the domain of $M \cup M_+$, $R(\mathbf{t}) \leftarrow \varrho$ contains an occurrence of a variable connected to the occurrence $h(\vartheta(x))$. We say that the pair (g, s), with g a node of τ and s a state of $\mathbf{A}^{\gamma}_{Q,\Pi}$, is *accepting* (for $\mathbf{A}^{\gamma}_{Q,\Pi}$ and τ) if there is an accepting run ν of $\mathbf{A}^{\gamma}_{Q,\Pi}$ on τ and a node f of ν such that $\nu(f) = (g, s)$.

We further say that M_+ is *consistent* with $h(f)$, if for each variable $\bar{v}^i_E$ in the domain of M_+ we have that M_+ maps $\bar{v}^i_E$ to $h(f)(\vartheta(\bar{v}^i_E))$.

We will inductively construct W and h such that, for each node f of ν (and hence of h, and W), such that $\nu(f)$ is accepting, we have that M and M_+ are consistent with $h(f)$ and that f is connected.

- Base case ("final" transition): If there is a "final" transition from a node f to a node $f{\cdot}1$ of ν that is a leaf of the run labeled with the accepting state, then $\beta = \emptyset$. Then $W(f)$ is empty and so is $h(f)$. Hence, trivially, $h(f)$ is a strong containment mapping from $W(f)$ to τ, M and M_+ are consistent with $h(f)$, and f is connected.
- Inductive case 1 ("atom mapping" transition): If there is an "atom mapping" transition from a node f to a node $f{\cdot}1$ of ν, with $\nu(f{\cdot}1) = (g, (R(\mathbf{t}), \beta', M, M'_+))$ accepting, then, by inductive hypothesis, $W(f{\cdot}1)$ consists of one sequence of atoms for each element $E^F_s(x, y)$ of β', $h(f{\cdot}1)$ is a strong containment mapping from $W(f{\cdot}1)$ to atoms in τ, M and M'_+ are consistent with $h(f{\cdot}1)$, and f is connected.

  We have that β coincides with β', except that one element $E^f_s(x, y)$ in β is replaced in β' by $E^F_{s'}(x', y)$. We consider only the case where there is an EDB atom $e(a, b)$ among the atoms in ϱ such that $s' \in \delta_E(s, e)$, and $x \in V_\gamma$. The other cases are similar. Then $W(f)$ is equal to $W(f{\cdot}1)$, except that the sequence $r^1(\vartheta(x'), x^1) \cdots r^n(x^{n-1}, \vartheta(y))$ of atoms corresponding to $E^F_{s'}(x', y)$ is replaced in $W(f)$ by $e(\vartheta(x), \vartheta(x')), r^1(\vartheta(x'), x^1) \cdots r^n(x^{n-1}, \vartheta(y))$ corresponding to $E^F_s(x, y)$.

  We extend $h(f{\cdot}1)$ to $h(f)$ by mapping the current occurrence of $\vartheta(x)$ to a. Observe that, if there are other occurrences of $\vartheta(x)$ in $h(f{\cdot}1)$ they are

mapped to a as well since M is consistent with $h(f{\cdot}1)$ and $f{\cdot}1$ is connected. Moreover, $M_+ = M'_+ \setminus \{(\bar{v}^1_E, b)\}$. Hence, $h(f)$ is a strong containment mapping from $W(f)$ to atoms in τ and M and M_+ are consistent with $h(f)$. Moreover since the transition stays in the same node and $\vartheta(x)$ is mapped to a variable in ϱ, we have that f is connected.

- Inductive case 2 ("splitting" transition): If there is a "splitting" transition from a node f to nodes $f{\cdot}1$ and $f{\cdot}2$ of ν with $\nu(f{\cdot}1) = (g, (R(\mathbf{t}), \beta', M, M'_+))$ and $\nu(f{\cdot}2) = (g, (R(\mathbf{t}), \beta'', M, M''_+))$ accepting, then, by inductive hypothesis, $W(f{\cdot}1)$ consists of one sequence of atoms for each element $E^F_s(x, y)$ of β', $h(f{\cdot}1)$ is a strong containment mapping from $W(f{\cdot}1)$ to atoms in τ, M and M'_+ are consistent with $h(f{\cdot}1)$, and $f{\cdot}1$ is connected; similarly for $f{\cdot}2$.

  We have that β coincides with $\beta' \cup \beta''$, except for elements $E^f_s(x, y)$ in β that are replaced in β' by $E^{s'}_s(x, \bar{v}^2_E)$ and in β'' by $E^F_{s'}(\bar{v}^1_E, y)$. (The case where $E^{s'}_s(x, \bar{v}^2_E)$ is in β'' and $E^F_{s'}(\bar{v}^1_E, y)$ is in β'' is analogous.) Then $W(f)$ is equal to $W(f{\cdot}1) \cup W(f{\cdot}2)$, except that, for each such $E^f_s(x, y)$ we have a sequence $w(z^{u-1}, z^{i-1}), w(z^{i-1}, z^v)$, where $w(z^{u-1}, z^{i-1}) = r^u(z^{u-1}, z^u), \ldots, r^{i-1}(z^{i-2}, z^{i-1})$ is the sequence of atoms in $W(f{\cdot}1)$ corresponding to $E^{s'}_s(x, \bar{v}^2_E)$ and $w(z^{i-1}, z^v) = r^i(z^{i-1}, z^i), \ldots, r^v(z^{v-1}, z^v)$ is the sequence in $W(f{\cdot}2)$ corresponding to $E^F_{s'}(varv^1_E, y)$.

  We take $h(f) = h(f{\cdot}1) \cup h(f{\cdot}2)$. Observe that, since $h(f{\cdot}1)$ and $h(f{\cdot}2)$ are both compatible with M and respectively compatible with M'_+ and M''_+, both occurrences of z^{i-1} in $w(z^{u-1}, z^{i-1})$ and in $w(z^{i-1}, z^v)$ are mapped to the same variable of ϱ. Hence, $h(f)$ is a strong containment mapping from $W(f)$ to atoms in τ and M and M_+ are consistent with $h(f)$. Moreover, since both conjuncts of the transition stay in the same node and z^{i-1} is mapped to a variable in ϱ, f is connected.
- Inductive case 3 ("moving" transition): If there is a "downward moving" (resp., "upward moving") transition from a node f to a node $f{\cdot}1$ of ν, with $\nu(f{\cdot}1) = (g{\cdot}j, (R_{i_j}(\mathbf{t}^{i_j}), \beta, M, M_+))$ (resp., $\nu(f{\cdot}1) = (g{\cdot}(-1), (R'(\mathbf{t}'), \beta, M, M_+))$) accepting, then, by inductive hypothesis, $W(f{\cdot}1)$ consists of one sequence of atoms for each element $E^F_s(x, y)$ of β, $h(f{\cdot}1)$ is a strong containment mapping from $W(f{\cdot}1)$ to atoms in τ, M and M_+ are consistent with $h(f{\cdot}1)$, and $f{\cdot}1$ is connected.

  We take $W(f) = W(f{\cdot}1)$ and $h(f) = h(f{\cdot}1)$. Trivially $h(f{\cdot}1)$ is a strong containment mapping from $W(f)$ to atoms in τ and M and M_+ are consistent with $h(f)$. Moreover, since all variables that occur in β and that are in the domain of $M \cup M_+$ have their image in $\mathbf{t}^{i_j}$ (resp., $\mathbf{p}'$), f is connected.
- Inductive case 4 ("equality checking" transition): If there is an "equality checking" transition from a node f to a node $f{\cdot}1$ of ν, with $\nu(f{\cdot}1) = (g, (R(\mathbf{t}), \beta', M, M_+))$ accepting, then, by inductive hypothesis, $W(f{\cdot}1)$ consists of one sequence of atoms for each element $E^F_s(x, y)$ of β, $h(f{\cdot}1)$ is a strong containment mapping from $W(f{\cdot}1)$ to atoms in τ, M and M_+ are consistent with $h(f{\cdot}1)$, and $f{\cdot}1$ is connected.

  For each atom $E^F_s(x, y)$ in β_0 we have that $s \in F$ and (x, a) and (y, a) are in $M \cup M_+$, for some variable a in $R(\mathbf{t}) \leftarrow \varrho$. Then $W(f)$ extends $W(f{\cdot}1)$ by

adding an equality atom $\vartheta(x) = \vartheta(y)$, and $h(f)$ extends $h(f{\cdot}1)$ by mapping the occurrence of $\vartheta(x)$ and of $\vartheta(y)$ to a. Hence, $h(f)$ is a strong containment mapping from $W(f)$ to atoms in τ and M and M_+ are consistent with $h(f)$. Moreover since the transition stays in the same node and $\vartheta(x)$ and $\vartheta(y)$ are mapped to a variable in $R(\mathbf{t}) \leftarrow \varrho$, we have that f is connected.

- Inductive case 5 ("mapping extending" transition): If there is a "mapping extending" transition from a node f to a node $f{\cdot}1$ of ν, with $\nu(f{\cdot}1) = (g, (R(\mathbf{t}), \beta, M', M_+))$ accepting, then, by inductive hypothesis, $W(f{\cdot}1)$ consists of one sequence of atoms for each element $E_s^F(x, y)$ of β, $h(f{\cdot}1)$ is a strong containment mapping from $W(f{\cdot}1)$ to atoms in τ, M and M_+ are consistent with $h(f{\cdot}1)$, and $f{\cdot}1$ is connected.

  We take $W(f) = W(f{\cdot}1)$ and $h(f) = h(f{\cdot}1)$. Trivially $h(f{\cdot}1)$ is a strong containment mapping from $W(f)$ to atoms in τ and M and M_+ are consistent with $h(f)$. Moreover, f is trivially connected since the transition stays in the same node, β remains the same, and M is smaller than M'.

Since in the initial state of $\mathbf{A}^{\gamma}_{Q,\Pi}$ we have that $\beta = \gamma$, we have $\varphi = W(\varepsilon)$ is an expansion of γ, and $h = h(\varepsilon)$ is a strong containment mapping from φ to τ. The claim follows. □

Theorem 11 *Let Π be a Datalog program with binary EDB predicates in Δ and with goal predicate Q, and let $\Gamma = \bigcup_i \gamma_i$ be a finite union of C2RPQs γ_i over $\Delta^{\pm}$. Then Π is contained in Γ if and only if*

$$\mathcal{T}(\mathbf{A}^{p\_trees}_{Q,\Pi}) \subseteq \bigcup_i \mathcal{T}(\mathbf{A}^{\gamma_i}_{Q,\Pi})$$

Proof. By Lemma 3, Π is contained in Γ if and only if for every proof tree $\tau \in p\_trees(Q, \Pi)$ there is a γ_i and an expansion φ of γ_i such that there is a strong containment mapping from φ to τ. By Theorem 8 and Lemmas 9 and 10, the latter conditions is equivalent to $\mathcal{T}(\mathbf{A}^{p\_trees}_{Q,\Pi}) \subseteq \bigcup_i \mathcal{T}(\mathbf{A}^{\gamma_i}_{Q,\Pi})$. □

This allows us to establish the main result of the paper.

Theorem 12 *Containment of a (recursive) Datalog program in a union of C2RPQs is in 2EXPTIME.*

Proof. By Proposition 5, we can construct a 2ATA $\mathbf{A}^{\Gamma}_{Q,\Pi}$, whose size is exponential in the size of Π and Γ, such that $\mathcal{T}(\mathbf{A}^{\Gamma}_{Q,\Pi}) = \bigcup_i \mathcal{T}(\mathbf{A}^{\gamma_i}_{Q,\Pi})$. By Proposition 4, we can construct a 1NTA $\mathbf{A}^{\neg\Gamma}_{Q,\Pi}$, whose size is doubly exponential in the size of Π and Γ, such that a Σ-labeled tree is accepted by $\mathbf{A}^{\neg\Gamma}_{Q,\Pi}$ if and only if it is not accepted by $\mathbf{A}^{\Gamma}_{Q,\Pi}$. By Proposition 6, we can construct a 1NTA $\mathbf{A}_{cont}$, whose size is still doubly exponential in the size of Π and Γ, such that $\mathbf{A}_{cont}$ accepts a Σ-labeled tree if and only if it is accepted by $\mathbf{A}^{p\_trees}_{Q,\Pi}$ but not accepted by any of the $\mathbf{A}^{\gamma_i}_{Q,\Pi}$. By Theorem 11, $\mathbf{A}_{cont}$ is nonempty if and only

if Π is not contained in Γ. By Proposition 7, nonemptiness of $\mathbf{A}_{cont}$ can be checked in time polynomial in its size, and hence doubly exponential in the size of Π and Γ. The claim follows. □

6 Lower Bound

Next we turn to the lower bound for containment of Datalog in unions of C2RPQs. In [25], it is shown that containment of Datalog in unions of conjunctive queries is 2EXPTIME-hard, by a reduction from acceptance of an alternating EXPTIME Turing machine. The encoding in that proof uses EDB predicates of arity different from 2, and hence does not directly apply to containment of Datalog in unions of C2RPQs, where all EDB predicates are binary. Nevertheless, the problem of containment of a Datalog program in a union of conjunctive queries over arbitrary EDB predicates can be reduced to the problem of containment of a Datalog problem in a union of conjunctive queries over binary EDB predicates, as shown below.

Let Π be a Datalog program with goal predicate Q over EDB predicates of arbitrary arity, and Φ a union of conjunctive queries over the EDB predicates of Π. We construct a Datalog program Π' with goal predicate Q over binary EDB predicates and a union of conjunctive queries Φ' over binary EDB predicates as follows:

- For each EDB predicate R of arity $n > 2$ appearing in Π or Φ we consider R in Π' as an IDB predicate, and we introduce n fresh binary EDB predicates R_i, for $i \in 1, \ldots, n$, which represent the components of tuples of R. Also, the following rule is added to Π':

  $$R(x_1, \ldots, x_n) \leftarrow R_1(y, x_1), \ldots, R_n(y, x_n)$$

  where y is an existential variable that represents the tuple $(x_1, \ldots, x_n)$.
- For each unary EDB predicate R appearing in Π or Φ, we consider R in Π' as an IDB predicate, and we introduce a fresh binary EDB predicate R_u. Also, the following rule is added to Π':

  $$R(x) \leftarrow R_u(x, x)$$

- For each 0-ary EDB predicate R appearing in Π or Φ, we consider R in Π' as an IDB predicate, and we introduce a fresh binary EDB predicate R_0. Also, the following rule is added to Π':

  $$R \leftarrow R_0(x, x)$$

- Π' additionally contains all rules of Π.

In the following, we call the binary EDB predicates R_i (resp., R_u or R_0) newly introduced in Π' *fresh* EDB predicates. The union of conjunctive queries Φ' is obtained from Φ by

- replacing each atom $R(z_1, \ldots, z_n)$ over an n-ary (with $n > 2$) predicate R with the conjunction of atoms $R_1(w, z_1), \ldots, R_n(w, z_n)$, where w is a fresh variable;
- replacing each unary atom $R(z)$ with the binary atom $R_u(z, z)$;
- replacing each 0-ary atom R with the binary atom $R_0(w, w)$, where w is a fresh variable.

Lemma 13 *Let Π be a Datalog program with goal predicate Q and Φ a union of conjunctive queries, both over arbitrary EDB predicates. Let Π' and Φ' be the Datalog program and the union of conjunctive queries, both over binary EDB predicates, defined from Π and Φ as above. Then Π is contained in Φ if and only if Π' is contained in Φ'.*

Proof. "$\Rightarrow$" Assume that for each expansion tree τ in $trees(Q, \Pi)$ there is a containment mapping from some conjunctive query in Φ to τ. We show that for each expansion tree τ' in $trees(Q, \Pi')$ there is a containment mapping from some conjunctive query in Φ' to τ'. Since each fresh EDB predicate appears in Π' only in the body of rules whose head is an EDB predicate of arity $n \neq 2$ of Π or φ, and such rules contain in their body only fresh EDB predicates, we have that each node in τ' containing a fresh EDB predicate is a leaf node. Moreover, there is an expansion tree τ in $trees(Q, \Pi)$ such that τ' is obtained from τ by adding for each node g of τ:

- for each EDB atom $R(x_1, \ldots, x_n)$, with $n > 2$, appearing in (the body of the rule instance labeling) g, a child of g labeled by a rule instance $R(x_1, \ldots, x_n) \leftarrow R_1(y, x_1), \ldots, R_n(y, x_n)$;
- for each unary EDB atom $R(x)$ appearing in g, a child of g labeled by a rule instance $R(x) \leftarrow R_u(x, x)$;
- for each 0-ary EDB atom R appearing in g, a child of g labeled by a rule instance $R \leftarrow R_0(x, x)$.

Let τ be an expansion tree in $trees(Q, \Pi)$. By hypothesis, there exists a containment mapping h from some conjunctive query φ in Φ to τ. Let φ' be the conjunctive query in Φ' obtained from φ. Consider an atom $R(z_1, \ldots, z_n)$, with $n > 2$, in φ, and let h map such an atom to an atom $R(x_1, \ldots, x_n)$ in a node g of τ. Let $R_1(w, z_1), \ldots, R_n(w, z_n)$ be the conjunction of atoms in φ' corresponding to $R(z_1, \ldots, z_n)$, where, by construction, w is a variable not appearing in any other atom of φ'. Consider the child g' of g in τ' corresponding to the expansion of $R(x_1, \ldots, x_n)$, and let g' be labeled by the rule instance $R(x_1, \ldots, x_n) \leftarrow R_1(y, x_1), \ldots, R_n(y, x_n)$. Then, we can extend h so that it maps w to y and the atoms containing w to the atoms in the body of the rule

instance labeling g'. Similarly, consider a unary atom $R(z)$ in φ, and let h map such an atom to an atom $R(x)$ in a node g of τ. Let $R_u(z,z)$ be the atom in φ' corresponding to $R(z)$. Consider the child g' of g in τ' corresponding to the expansion of $R(x)$, and let g' be labeled by the rule instance $R(x) \leftarrow R_u(x,x)$. Then, since h maps z to x, it also maps the atom $R_u(z,z)$ to $R_u(x,x)$, which is the only atom in the body of the rule instance labeling g'. We can proceed in a similar way for 0-ary atoms. It is immediate to verify that, by proceeding in the same way for all non-binary atoms of φ, we have that h is a containment mapping from φ' to τ'.

"$\Leftarrow$" Assume that for each expansion tree τ' in $trees(Q,\Pi')$ there is a containment mapping from some conjunctive query in Φ' to τ', and let τ be an expansion tree in $trees(Q,\Pi)$. We show that there is a containment mapping from some conjunctive query in Φ to τ. Let τ' be an expansion tree in $trees(Q,\Pi')$ obtained from τ by adding for each node g of τ and each EDB atom $R(x_1,\ldots,x_n)$, with $n > 2$, appearing in (the body of the rule instance labeling) g, a child of g labeled by a rule instance $R(x_1,\ldots,x_n) \leftarrow R_1(y,x_1),\ldots,R_n(y,x_n)$, where y is a different fresh variable for each atom. Similarly for each unary and 0-ary EDB atom appearing in a node of τ. By hypothesis, there is a conjunctive query φ' in Φ' such that there exists a containment mapping h from φ' to τ'. Let φ be the conjunctive query in Φ from which φ' is derived. Consider a conjunction of atoms $R_1(w,z_1),\ldots,R_n(w,z_n)$ in φ' corresponding to an atom $R(z_1,\ldots,z_n)$ in φ, where, by construction, w is a variable not appearing in any other atom of φ'. Let g' be the node of τ' containing (in the body of the rule instance labeling g') the atom $R_1(y,x_1)$ to which h maps $R_1(w,z_1)$. By construction of τ' the variable y appears only in atoms of g'. Hence, the rule instance labeling g' will be of the form $R(x_1,\ldots,x_n) \leftarrow R_1(y,x_1),\ldots,R_n(y,x_n)$, where $R_1(y,x_1),\ldots,R_n(y,x_n)$ are the atoms to which h maps $R_1(w,z_1),\ldots,R_n(w,z_n)$, respectively. It follows that we can map the atom $R(z_1,\ldots,z_n)$ in φ to the atom in the head $R(x_1,\ldots,x_n)$ of the rule instance labeling g', or, equivalently, to the atom $R(x_1,\ldots,x_n)$ in the predecessor node g of g' in τ' and hence also in τ. We can reason in a similar way for binary atoms in φ' corresponding to unary and 0-ary atoms of φ. It is immediate to verify that, by proceeding as above for all conjunctions of atoms in φ' corresponding to atoms of φ of arity greater than 2, and for all atoms in φ' corresponding to unary and 0-ary atoms of φ, we have that h is a containment mapping from φ to τ. □

Considering that the construction above is linear in Π and φ, from 2EXPTIME-hardness of containment of Datalog in unions of conjunctive queries over arbitrary EDB predicates [25], we obtain the following result.

Theorem 14 *Containment of a Datalog program in a union of conjunctive queries, both over binary EDB predicates, is 2EXPTIME-hard.*

By Theorem 12, we get the following computational complexity characterization.

Theorem 15 *Containment of a (recursive) Datalog program in a union of C2RPQs is 2EXPTIME-complete.*

7 Conclusions

We have established decidability of containment of Datalog queries in unions of conjunctive 2-way regular path queries, and characterized the complexity of the problem as 2EXPTIME-complete. This is the most general known decidability result for containment of recursive queries, apart from the result in [23] for monadic Datalog. The class of union of C2RPQs has several features that are typical of modern query languages, in particular of those for semistructured data. Unions of C2RPQs constitute the largest fragment of query languages for XML data [45] for which containment is known to be decidable [34].

The 2EXPTIME upper-bound result shows that adding transitive closure to conjunctive queries does not increase the complexity of query containment with respect to Datalog queries, as it matches the bound obtained in [25] for containment of Datalog queries in union of conjunctive queries. Observe that containment in the converse direction, as well as equivalence, is undecidable already for RPQs. Indeed, universality of context free grammars can be reduced to containment of RPQs in Datalog, by following the line of the undecidability proof of containment between Datalog queries in [22].

Query containment is typically the first step in addressing various problems of query processing, such as view-based query processing. One of the most important view-based query processing tasks is *view-based query answering* [46,47], where one is interested in computing the answer to a query over a global virtual schema, based on the data stored in a set of materialized views, defined also over the virtual schema. In such a setting, the typical assumption is that views are *sound*, i.e., the data available in the views are a subset of the data satisfying the corresponding view definition [47]. There is a well-known connection between query containment and view-based query answering (under sound views) [48,49], that is based on using the data in the views to construct the body of the query on the left-hand side of containment. By exploiting such a connection [4], the results in this paper already show that view based

[4] The reductions between query containment and view-based query answering in [48] make use of constants in the query built from the views. However, it is easy to see that, since we do not allow for inequalities in queries, the reductions can

query answering is decidable and 2EXPTIME-complete when the views are Datalog and the query is a union of C2RPQs. This is the most general known decidability result for view-based query answering in the presence of recursion.

References

[1] M. Buchheit, M. A. Jeusfeld, W. Nutt, M. Staudt, Subsumption between queries to object-oriented databases, Information Systems 19 (1) (1994) 33–54, special issue on Extending Database Technology, EDBT'94.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003.

[3] A. Gupta, J. D. Ullman, Generalizing conjunctive query containment for view maintenance and integrity constraint verification (abstract), in: Workshop on Deductive Databases (In conjunction with JICSLP), Washington D.C. (USA), 1992, p. 195.

[4] A. Y. Levy, Y. Sagiv, Semantic query optimization in Datalog programs, in: Proc. of the 14th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'95), 1995, pp. 163–173.

[5] S. Chaudhuri, S. Krishnamurthy, S. Potarnianos, K. Shim, Optimizing queries with materialized views, in: Proc. of the 11th IEEE Int. Conf. on Data Engineering (ICDE'95), 1995.

[6] S. Adali, K. S. Candan, Y. Papakonstantinou, V. S. Subrahmanian, Query caching and optimization in distributed mediator systems, in: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 1996, pp. 137–148.

[7] P. Buneman, S. Davidson, G. Hillebrand, D. Suciu, A query language and optimization technique for unstructured data, in: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 1996, pp. 505–516.

[8] A. Motro, Panorama: A database system that annotates its answers to queries with their properties, J. of Intelligent Information Systems 7 (1).

[9] A. Y. Levy, M.-C. Rousset, Verification of knowledge bases: a unifying logical view, in: Proc. of the 4th European Symposium on the Validation and Verification of Knowledge Based Systems, Leuven, Belgium, 1997.

[10] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati, Description logic framework for information integration, in: Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98), 1998, pp. 2–13.

be rephrased without making use of constants.

[11] M. F. Fernandez, D. Florescu, A. Levy, D. Suciu, Verifying integrity constraints on web-sites, in: Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99), 1999, pp. 614–619.

[12] M. Friedman, A. Levy, T. Millstein, Navigational plans for data integration, in: Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI'99), AAAI Press/The MIT Press, 1999, pp. 67–73.

[13] T. Milo, D. Suciu, Index structures for path expressions, in: Proc. of the 7th Int. Conf. on Database Theory (ICDT'99), Vol. 1540 of Lecture Notes in Computer Science, Springer, 1999, pp. 277–295.

[14] A. K. Chandra, P. M. Merlin, Optimal implementation of conjunctive queries in relational data bases, in: Proc. of the 9th ACM Symp. on Theory of Computing (STOC'77), 1977, pp. 77–90.

[15] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison Wesley Publ. Co., Reading, Massachussetts, 1995.

[16] A. K. Chandra, D. Harel, Horn clause queries and generalizations, J. of Logic and Computation 2 (1985) 1–15.

[17] Y. N. Moschovakis, Elementary Induction on Abstract Structures, North-Holland Publ. Co., Amsterdam, 1974.

[18] A. V. Aho, Y. Sagiv, J. D. Ullman, Equivalence among relational expressions, SIAM J. on Computing 8 (1979) 218–246.

[19] Y. Sagiv, M. Yannakakis, Equivalences among relational expressions with the union and difference operators, J. of the ACM 27 (4) (1980) 633–655.

[20] A. C. Klug, On conjunctive queries containing inequalities, J. of the ACM 35 (1) (1988) 146–160.

[21] R. van der Meyden, The complexity of querying indefinite information, Ph.D. thesis, Rutgers University (1992).

[22] O. Shmueli, Equivalence of Datalog queries is undecidable, J. of Logic Programming 15 (3) (1993) 231–241.

[23] S. S. Cosmadakis, H. Gaifman, P. C. Kanellakis, M. Y. Vardi, Decidable optimization problems for database logic programs, in: Proc. of the 20th ACM SIGACT Symp. on Theory of Computing (STOC'88), 1988, pp. 477–490.

[24] Y. Sagiv, Optimizing Datalog programs, in: J. Minker (Ed.), Foundations of Deductive Databases and Logic Programming, Morgan Kaufmann, Los Altos, 1988, pp. 659–698.

[25] S. Chaudhuri, M. Y. Vardi, On the equivalence of recursive and nonrecursive datalog programs, J. of Computer and System Sciences 54 (1) (1997) 61–78.

[26] S. Chaudhuri, M. Y. Vardi, On the complexity of equivalence between recursive and nonrecursive Datalog programs, in: Proc. of the 13th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'94), 1994, pp. 107–116.

[27] T. Bray, J. Paoli, C. M. Sperberg-McQueen, Extensible Markup Language (XML) 1.0 — W3C recommendation, Tech. rep., World Wide Web Consortium, available at `http://www.w3.org/TR/1998/REC-xml-19980210` (1998).

[28] D. Calvanese, G. De Giacomo, M. Lenzerini, Representing and reasoning on XML documents: A description logic approach, J. of Logic and Computation 9 (3) (1999) 295–318.

[29] P. Buneman, Semistructured data, in: Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97), 1997, pp. 117–121.

[30] D. Florescu, A. Levy, A. Mendelzon, Database techniques for the World-Wide Web: A survey, SIGMOD Record 27 (3) (1998) 59–74.

[31] S. Abiteboul, P. Buneman, D. Suciu, Data on the Web: from Relations to Semistructured Data and XML, Morgan Kaufmann, Los Altos, 2000.

[32] S. Abiteboul, V. Vianu, Regular path queries with constraints, J. of Computer and System Sciences 58 (3) (1999) 428–452.

[33] D. Florescu, A. Levy, D. Suciu, Query containment for conjunctive queries with regular expressions, in: Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98), 1998, pp. 139–148.

[34] D. Calvanese, G. De Giacomo, M. Lenzerini, M. Y. Vardi, Containment of conjunctive regular path queries with inverse, in: Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000), 2000, pp. 176–185.

[35] G. Slutzki, Alternating tree automata, Theoretical Computer Science 41 (1985) 305–318.

[36] D. Maier, J. D. Ullman, M. Y. Vardi, On the foundations of the universal relation model, ACM Trans. on Database Systems 9 (1984) 283–308.

[37] J. F. Naughton, Data independent recursion in deductive databases, J. of Computer and System Sciences 38 (2) (1989) 259–289.

[38] J. E. Hopcroft, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison Wesley Publ. Co., Reading, Massachussetts, 1979.

[39] F. Gecseg, M. Steinby, Tree Automata, Akademiai Kiado, 1984.

[40] J. E. Doner, Tree acceptors and some of their applications, J. of Computer and System Sciences 4 (5) (1970) 406–451.

[41] J. W. Thatcher, J. B. Wright, Generalized finite automata theory with an application to a decision problem of second order logic, Mathematical Systems Theory 2 (1) (1968) 57–81.

[42] O. L. Costich, A Medvedev characterization of sets recognized by generalized finite automata, Mathematical Systems Theory 6 (1972) 263–267.

[43] C. Beeri, P. A. Bernstein, Computational problems related to the design of normal form relational schemas, ACM Trans. on Database Systems 4 (1) (1979) 30–59.

[44] W. F. Dowling, J. H. Gallier, Linear-time algorithms for testing the satisfiability of propositional horn formulae, J. of Logic Programming 1 (3) (1984) 267–284.

[45] A. Deutsch, M. F. Fernandez, D. Florescu, A. Levy, D. Maier, D. Suciu, Querying XML data, Bull. of the IEEE Computer Society Technical Committee on Data Engineering 22 (3) (1999) 10–18.

[46] A. Y. Halevy, Answering queries using views: A survey, Very Large Database J. 10 (4) (2001) 270–294.

[47] M. Lenzerini, Data integration: A theoretical perspective., in: Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002), 2002, pp. 233–246.

[48] S. Abiteboul, O. Duschka, Complexity of answering queries using materialized views, in: Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98), 1998, pp. 254–265.

[49] D. Calvanese, G. De Giacomo, M. Lenzerini, M. Y. Vardi, View-based query answering and query containment over semistructured data, in: G. Ghelli, G. Grahne (Eds.), Revised Papers of the 8th International Workshop on Database Programming Languages (DBPL 2001), Vol. 2397 of Lecture Notes in Computer Science, Springer, 2002, pp. 40–61.