Boundedness of Regular Path Queries in Data Integration Systems

Gösta Grahne
Concordia University
QC, Canada
grahne@cs.concordia.ca

Alex Thomo
University of Victoria
BC, Canada
thomo@cs.uvic.ca

Abstract

In this paper we study the problem of deciding whether a regular path query over views in data-integration systems can be re-expressed without recursion. The problem becomes challenging when the views contain recursion, thereby potentially making recursion in the query uncessary. We define two related notions of boundedness of regular path queries. For one of the notions we show it PSPACE complete, and obtain a constructive method for optimizing regular path queries in data-integration systems. For the other notion of boundedness, we show it PTIME reducible to the notorius problem of limitedness in distance automata, for which only exponential time algorithms are currently known.

1 Introduction

The compile time query optimization is one of the key factors for the enormous success of database systems today. Notably, the majority of the influential work on query optimizers delt with SQL queries, which correspond to datalog queries without recursion.

Nevertheless, in the research community from the mid 1980's to the mid 1990's, another theme was the study of (recursive) Datalog (see *e. g.* [27]). Unfortunately, most decision problems related to query optimization turned out to be undecidable. One of these undecidable problems was the boundedness of Datalog, which is to decide whether a given recursive Datalog query is equivalent with one without recursion. The importance of this is that should we be able to re-express a recursive Datalog query as another query without recursion, then we could use the optimization machinery for non-recursive queries, which over the years has been proven to be very efficient and successful in commercial systems.

Notably, a well-behaved fragment of Datalog, which is both natural and quite general, did emerge in the mid 1990s, in the context of semistructured data (graph data) (see [27]). This fragment is the class of regular queries, whose basic element is that of regular path queries.

The semi-structured data model [1] is now widely used as a foundation for reasoning about a multitude of applications, where the data is best formalized in terms of labeled graphs. Such data is usually found in Web information systems, XML data repositories, digital libraries, communication networks, and so on.

Regarding the query languages for semi-structured data, virtually all of them provide the possibility for the user to query the database through regular expressions. Fortunately, the boundedness problem for regular path queries is decidable. Simply, one has to build a finite automaton and check whether there is a cycle on a path between an initial and a final state.

This simplicity is not true anymore when the regular path queries are on an alphabet where the symbols represent views, which can in turn be recursive. Such queries are prominent today in information integration systems, where the data-sources are represented as a set of views over a *global schema*. The data is described by the *local schema*, which in the semi-structured context is the set of view names. By using the view definitions, the user query (expressed on the global schema) is rewritten in terms of the local schema. Finally, the obtained view-based rewriting is used to extract the answer from the data on the local schema. This is commonly referred in the literature as the *local-as-view* (LAV) approach for data integration (see [18]).

When it comes to decide whether a view-based rewriting is bounded, the above-mentioned simple check for recursion is not sufficient. A classical example presented in [3] is the following. Suppose that we have a single view $V = R^*$ and the query or the view-based rewriting is V^*. Clearly, this is equivalent with just V, which is more efficient than V^* to be answered on the data source.

We note here that, the view-based rewritings, generated by the method proposed in [3] always contain *all* the recursion possible, as long as the containment of the rewriting to the original query is preserved. The problem of "minimizing" a rewriting was first proposed in the same paper ([3])

but it has been open since then.

In general, the problem is more complicated than the example above, which illustrates only a case where the non-recursive rewriting reduces to a single letter word. As a more elaborated example, consider the view $V = R^+$. Now, for any given (natural) number k, take the query $Q = R^*.R^k$. The rewriting computed by [3] and [8] will be $(V^k)^+$, and clearly we need to have a way to infer that in fact the only word needed from this language is the word V^k, which has length k.

In this paper, by solving the boundedness problem for regular path queries over views, we show that for the *exact* view-based assumption in data integration, it is sometimes possible to replace a view-based rewriting with a not necessarily (purely) algebraically equivalent non-recursive one, without loosing any answers. The exact view assumption is usually the implicit assumption in a multitude of applications such as datawarehouses and enterprise data integration applications (see e.g. [11]), and has received considerable attention in the research community (see e.g. [6, 2]).

Furthermore, we obtain an optimal algorithm, which takes as input a view-based expression and a number k, and returns an equivalent expression without recursion (if such exists), in which the length of the longest word does not exceed k.

Depending on the application, we might be interested only in the *existence* of the above number k. Namely, we would like to know, for a given expression on the view definitions, whether there exists a number k, such that the sublanguage of the words of length not more than k is equivalent with the language captured by the original expression. Clearly, this amounts to deciding whether a query can be equivalently re-expressed without recursion. We show that our existential problem is polynomial time reducible to the intricate limitedness problem for distance automata, intensely investigated by Hashiguchi and others [13, 14, 15, 19, 25, 23].

2 Basic Definitions

We consider a database to be an edge labeled graph. This graph model is typical in semistructured data, where the nodes of the database graph represent the objects and the edges represent the attributes of the objects, or relationships between the objects.

Formally, let Δ be a finite alphabet. We shall call Δ the *database alphabet*. Elements of Δ will be denoted $R, S, \ldots$. As usual, Δ^* denotes the set of all finite words over Δ. Words will be denoted by $u, w, \ldots$. We also assume that we have a universe of objects, and objects will be denoted $a, b, c, \ldots$. A *database* $\mathcal{D}$ over Δ is a is a subset of $N \times \Delta \times N$, where N is a finite set of objects, that we usually will call nodes. We view a database as a directed labeled graph, and interpret a triple (a, R, b) as a directed edge from a to object b, labeled with R. If there is a path labeled $R_1, R_2, \ldots, R_k$ from a node a to a node b we write $a \xrightarrow{R_1 R_2 \ldots R_k} b$.

A *(user) query* Q is a regular language over Δ. For the ease of notation, we will blur the distinction between regular languages and regular expressions that represent them. Let Q be a query and $\mathcal{D}$ a database. Then, the *answer* to Q on $\mathcal{D}$ is defined as

$$ans(Q, \mathcal{D}) = \{(a, b) : a \xrightarrow{w} b \text{ in } \mathcal{D} \text{ for some } w \in Q\}.$$

Let $\Omega = \{v_1, \ldots, v_n\}$ be an *outer alphabet*, frequently also called a *view alphabet*. A *view graph* is database $\mathcal{V}$ over Ω. In other words, a view graph is a database where the edges are labeled with symbols from Ω. View graphs can also be queried (by queries over Ω), and $ans(Q, \mathcal{V})$ is defined like the answer to Q on a database $\mathcal{D}$, *mutatis mutandis*.

Let h be a homomorphism from Ω to subsets of Δ^*, *i. e.* for each v_i, $h(v_i)$ is a finite or infinite regular language over Δ. The homomorphism h is applied to words, languages, and regular expressions in the usual way (see *e. g.* [16]). We shall often denote the languages $h(v_i)$ with V_i and call them *views*.

In a LAV information integration system [18], we have the "global schema" Δ, the "source schema" Ω, and the "assertion" $h : \Omega \rightarrow 2^{\Delta^*}$. The only extensional data available is a view graph $\mathcal{V}$ over Ω (see also [20, 26, 7, 5][1]). The user queries are expressed on the global schema Δ, and the system has to reason about the information it can extract from the view graph $\mathcal{V}$. In order to do this, it has to consider the set of *possible databases* over Δ that $\mathcal{V}$ could represent. Under the *exact view* assumption, a view graph $\mathcal{V}$ defines in the information integration system a set $poss(\mathcal{V})$ of databases as follows:

$$poss(\mathcal{V}) = \{\mathcal{D} : \mathcal{V} = \bigcup_{i \in \{1, \ldots, n\}} \{(a, v_i, b) : (a, b) \in ans(V_i, \mathcal{D})\}\}.$$

(Recall that $V_i = h(v_i)$.[2]) The above definition reflects the intuition about the connection between an edge (a, v_i, b) in $\mathcal{V}$ with the set of paths between a and b in the possible $\mathcal{D}$'s, labeled by some word in V_i. The meaning of querying a view graph through the global schema in a LAV information integrations system is defined as follows. Let Q be a query over Δ. Then

$$\text{ANS}(Q, \mathcal{V}) = \bigcap_{\mathcal{D} \in poss(\mathcal{V})} ans(Q, \mathcal{D}).$$

[1]Regarding corresponding LAV scenarios for relational data.

[2]Furthermore, if we replace the equality sign for $\mathcal{V}$ with "$\subseteq$" then the views are considered to be *sound*.

Henceforth, we will consider only view graphs which are *valid*, that is, the view graphs for which the set of possible databases is not empty. Under the exact view assumption, not all view graphs are valid. As an example, consider a single view $V = R^*$, and the view graph $\mathcal{V} = \{(a, v, b), (b, v, c)\}$. It is easy to see that $poss(\mathcal{V}) = \emptyset$. The reason is that $\mathcal{V}$ "misses" a v-edge from a to c.

There are two approaches for computing $\text{ANS}(Q, \mathcal{V})$. The first one is to use an exponential procedure in the size of the data in order to completely compute $\text{ANS}(Q, \mathcal{V})$ (see [4]). There is little that one can better hope for, since in the same paper it has been proven that to decide whether a tuple belongs to $\text{ANS}(Q, \mathcal{V})$ is co-NP complete with respect to the size of data.

The second approach is to compute first a view-based rewriting Q' for Q, as in [3]. Such rewritings are regular path queries on Ω. Then, we can approximate $\text{ANS}(Q, \mathcal{V})$ by $ans(Q', \mathcal{V})$, which can be computed in polynomial time with respect to the size of data. In general, for a view-based rewriting Q' computed by the algorithm of [3], we have that

$$ans(Q', \mathcal{V}) \subseteq \text{ANS}(Q, \mathcal{V}),$$

with equality when the rewriting is exact ([4]). In the rest of the paper, we will assume that the data-integration system follows the second approach.

3 Query Equivalence and Boundedness

Consider two queries, Q_1 and Q_2 over an alphabet $\Sigma \in \{\Delta, \Omega\}$. We say that a query Q_1 is Σ-*contained* in a query Q_2 denoted $Q_1 \subseteq_\Sigma Q_2$ iff the answer to Q_1 is contained to the answer to Q_2, on all databases over Σ. We say that Q_1 is Σ-*equivalent* to Q_2 and write $Q_1 \equiv_\Sigma Q_2$, when $Q_1 \subseteq_\Sigma Q_2$ and $Q_2 \subseteq_\Sigma Q_1$. It is easy to see that the above query containment coincides with the (algebraic) language containment of Q_1 and Q_2, and that the query equivalence coincides with the language equality, *i. e.* $Q_1 \subseteq_\Sigma Q_2$ iff $Q_1 \subseteq Q_2$ and $Q_1 \equiv_\Sigma Q_2$ iff $Q_1 = Q_2$.

Let Q_1 and Q_2 be queries over Ω. We say that Q_1 is Ω/Δ-*contained* in Q_2, denoted $Q_1 \subseteq_{\Omega/\Delta} Q_2$, iff $h(Q_1) \subseteq_\Delta h(Q_2)$. Likewise, Q_1 is Ω/Δ-*equivalent* to Q_2 denoted $Q_1 \equiv_{\Omega/\Delta} Q_2$, when $Q_1 \subseteq_{\Omega/\Delta} Q_2$ and $Q_2 \subseteq_{\Omega/\Delta} Q_1$. It is easy to see that Ω-containment $Q_1 \subseteq_\Omega Q_2$, implies Ω/Δ-containment $Q_1 \subseteq_{\Omega/\Delta} Q_2$ but not vice-versa. As an example, if $Q_1 = v$, $Q_2 = v^*$ (where $v \in \Omega$), and $h(v) = R^*$, then Q_1 is Ω/Δ-equivalent with Q_2, although they are not Ω-equivalent.

We now have the following theorem.

Theorem 1 *Let Q_1 and Q_2 be queries over Ω. Under the exact view assumption, $Q_1 \equiv_{\Omega/\Delta} Q_2$ iff for each valid view graph $\mathcal{V}$ over Ω, $ans(Q_1, \mathcal{V}) = ans(Q_2, \mathcal{V})$.*

The importance of this theorem is that it allows us to minimize as much as possible a query on Ω (i.e. a view-based rewriting) without loosing query-power as long as we preserve Ω/Δ-equivalence, which is algebraically weaker than Ω-equivalence.

The above does not hold when we drop the exactness assumption for the views and consider them sound only. As an example, consider a view V, which is Δ-equivalent with V^*, and a view graph $\mathcal{V} = \{(a, v, b), (b, v, c)\}$. For this $\mathcal{V}$, we have that $ans(v^*, \mathcal{V}) \neq ans(v, \mathcal{V})$. Clearly, the answer of V will be equal to the answer of V^* on each database on Δ, but because the view is assumed to be sound we cannot enforce $\mathcal{V}$ to have an additional v-edge from a to c.

We give now two definitions for the boundedness of a query Q on the Ω alphabet. For this, we denote with $Q^{(k)}$ the set of *all* words in Q, which have length of not more than k. Obviously, $Q^{(0)} \subseteq Q^{(1)} \subseteq \ldots \subseteq Q^{(k)} \subseteq \ldots \subseteq Q$.

Definition 1

1. *We say that Q is k-*bounded *iff $Q^{(k)} \equiv_{\Omega/\Delta} Q$.*
2. *We say that Q is* finitely bounded *iff there exists a $k \in \mathbb{N}$, such that Q is k-bounded.*

Although related, the problems of k-boundedness and finite boundedness are different. For the k-boundedness problem, the input is a query, and a fixed number k that the user provides. Then, the question is whether the query can semantically be fully represented by the Ω query words of length at most k.

On the other hand, for the finite boundedness problem, k is not part of the input, and the question is existential. Depending on the application we could be interested in the k or the finite boundedness.

As a first observation, the problem of k-bounded-ness is decidable. For this, we can (naively) enumerate from a query Q all the words of length at most k, thereby obtaining a bloated representation of $Q^{(k)}$, and then check $Q^{(k)} \equiv_{\Omega/\Delta} Q$ by testing for the algebraic language equivalence $h(Q^{(k)}) \equiv_\Delta h(Q)$. However, by naively enumerating words and then checking Ω/Δ-equivalence, we will get an exponential space penalty.

Even if we can decide the k-boundedness more efficiently, it is still important to have a concise representation of $Q^{(k)}$, since the purpose is to run query $Q^{(k)}$ instead of Q (after testing $Q^{(k)} \equiv_{\Omega/\Delta} Q$). As it turns out, our constructive proof of PSPACE-decidability for the k-boundedness problem also gives us an algorithm for building a concise automaton for $Q^{(k)}$ (see Theorem 4).

4 Weighted Transducers

In this section, we define weighted transducers, which will help us to solve both boundedness problems.

A *weighted transducer* $\mathcal{T} = (P, I, O, \tau, S, F)$ consists of a finite set of states P, an input alphabet I, an output alphabet O, a set of starting states S, a set of final states F, and a transition relation $\tau \subseteq P \times I^* \times O^* \times \mathbb{N} \times P$.

Given a weighted transducer $\mathcal{T}$ defined as above, and a word $u \in I^*$ we say that a word $w \in O^*$ is an *output of $\mathcal{T}$ for u through a k-weighted path* if there exists a sequence $(p_0, u_1, w_1, k_1, p_1), (p_1, u_2, w_2, k_2, p_2), \ldots, (p_{n-1}, u_n, w_n, k_n, p_n)$ of state transitions of τ, such that $p_0 \in S$, $p_n \in F$, $u = u_1 \ldots u_n$, $w = w_1 \ldots w_n$, and $k = k_1 + \cdots + k_n$.

We denote the set of all outputs of $\mathcal{T}$ for u (regardless of the path weight) by $\mathcal{T}(u)$. For a language $L \subseteq I^*$, we define $\mathcal{T}(L) = \bigcup_{u \in L} \mathcal{T}(u)$. We will also need the notation *rel*$(\mathcal{T})$ to denote the set of all pairs $(u, w) \in I^* \times O^*$, where w is an output of $\mathcal{T}$ when providing u as input. Similarly, *dom*$(\mathcal{T})$ and *ran*$(\mathcal{T})$, will be used to denote the domain and range of *rel*$(\mathcal{T})$.

Given a weighted transducer $\mathcal{T}$, and words u and w, the $\mathcal{T}$*-cost* for u to be translated into w is defined as

$$c_{\mathcal{T}}(u, w) = \begin{cases} \mathit{inf}\{k : w \text{ is an output of } \mathcal{T} \text{ for } u \\ \qquad \text{through a } k\text{-weighted path}\} \\ \infty, \text{ if } w \notin \mathcal{T}(u). \end{cases}$$

Now, we will define the $\mathcal{T}$-cost for translating one language into another. This will be needed in the formulation of a necessary and sufficient condition for the k- and the finite boundedness.

Consider a word w on I. The $\mathcal{T}$-cost for a word w to be translated into a language L_2, is

$$c_{\mathcal{T}}(w, L_2) = \mathit{inf}\{c_{\mathcal{T}}(w, u) : u \in L_2\}.$$

Based on that, the $\mathcal{T}$-cost for a language L_1 to be translated into a language L_2 can be naturally defined as

$$c_{\mathcal{T}}(L_1, L_2) = \mathit{sup}\{c_{\mathcal{T}}(w, L_2) : w \in L_1\}.$$

Returning to our problem, for a query Q on Ω, we construct an automaton $\mathcal{A} = (\{p_1, \ldots, p_m\}, \Omega, \tau, S, F)$ that recognizes Q (note that $S \subseteq \{p_1, \ldots, p_m\}$). Having the set $\{V_1, \ldots, V_n\}$ of view definitions, we also construct for each V_i, where $i \in [1, n]$, the (identical) automata $\mathcal{A}_{ijk} = (P_{ijk}, \Delta, \tau_{ijk}, S_{ijk}, F_{ijk})$, each recognizing V_i, whenever there is a transition $(p_j, v_i, p_k) \in \tau$, in $\mathcal{A}$, for $j, k \in [1, m]$.

Now, from the automata $\mathcal{A}$ and $\mathcal{A}_{ijk}$, for $i \in [1, n]$ and $j, k \in [1, m]$, we construct the transducer $\mathcal{T} = (P_{\mathcal{T}}, \Delta, \Omega, \tau_{\mathcal{T}}, S, F)$, where $P_{\mathcal{T}} = \{p_1, \ldots, p_m\} \cup \{\bigcup_{(p_j, v_1, p_k) \in \tau} P_{1jk}\} \cup \ldots \cup \{\bigcup_{(p_j, v_n, p_k) \in \tau} P_{njk})\}$, and

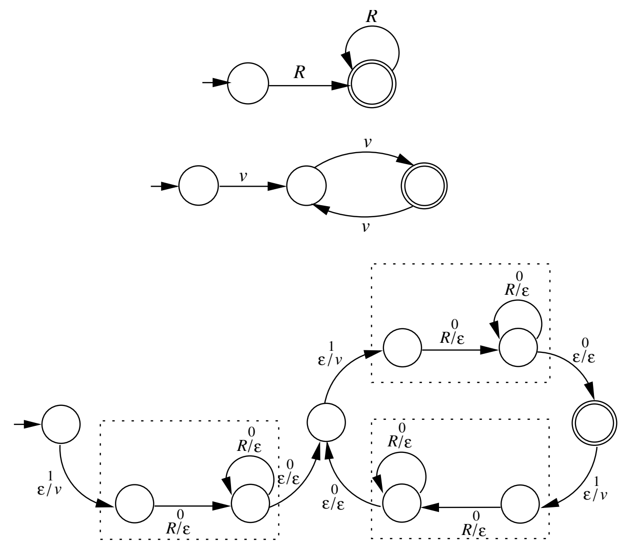

Figure 1. *Transducer construction*

$$\begin{aligned} \tau_{\mathcal{T}} \;=\; & \{(p_j, \epsilon, v_i, 1, s) : \\ & \qquad (p_j, v_i, p_k) \in \tau \text{ and } s \in S_{ijk}\} \cup \\ & \{(f, \epsilon, \epsilon, 0, p_k) : \\ & \qquad f \in F_{ijk} \text{ and } (p_j, v_i, p_k) \in \tau\} \cup \\ & \{(p, R, \epsilon, 0, q) : \\ & \qquad (p, R, q) \in \tau_{ijk} \text{ for some } i, j, k\}. \end{aligned}$$

The intuition behind the above construction is that, we replace the transitions of the query automaton $\mathcal{A}$ by the view automata corresponding to the transition labels, creating so, what might figuratively be called, view-automata "pockets." Whenever the transition "jumps" into some view-automaton "pocket" there is a cost penalty of one.

An example is given in Figure 1, in which we have a single view $V = R^+$ (top) and a query $Q = (v^2)^+$ (midle) on the alphabet $\Omega = \{v\}$.[3] The resulting transducer is shown at the bottom of the figure, where the view-automata "pockets" are surrounded by dotted rectangles. By the above construction, we get a weighted transducer $\mathcal{T}$, which has *dom*$(\mathcal{T}) = h(Q)$ and *ran*$(\mathcal{T}) = Q$. Also, the transducer $\mathcal{T}$ has the following property: It associates with each word $w \in h(Q)$ all the words $u \in Q$ such that $w \in h(u)$.

Now, we give the following characterization of the k- and the finite boundedness.

[3] The (machine generated) $Q = (v^2)^+$ is a view-based rewriting of some user query (e.g. $R^*.R^2$).

Theorem 2

1. *Q is k-bounded if and only if $c_{\mathcal{T}}(h(Q), Q) \leq k$.*

2. *Q is finitely bounded if and only if there is a $k \in \mathbb{N}$, such that $c_{\mathcal{T}}(h(Q), Q) \leq k$.*

In the next section we give an algorithm for testing whether $c_{\mathcal{T}}(\mathit{dom}(\mathcal{T}), \mathit{ran}(\mathcal{T})) \leq k$, for a given $k \in \mathbb{N}$.

5 Deciding k-Boundedness

Interestingly, deciding the k-boundedness bears some resemblance to deciding the query containment under distortions of [10]. Nevertheless, the constructions of [10] provide a mechanism for saying "yes" or "no" to a decision problem, while the constructions in this section, in addition to deciding the k-boundedness of view-based rewritings, also give a method for effectively obtaining the non-recursive equivalent rewriting if such exists. Furthermore, since it is not possible to reduce an arbitrary instance of the problem in [10] into our problem, we carefully examine and prove the complexity bounds for the k-boundedness of view-based rewritings.

We consider the transducer $\mathcal{T}$, constructed in the previous section. We will need a few simple operations on transducers. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be transducers. Then we denote with $\mathcal{T}_1 \cup \mathcal{T}_2$ the (union) transducer, obtained by the usual construction, translating $\mathit{dom}(\mathcal{T}_1) \cup \mathit{dom}(\mathcal{T}_2)$ into $\mathit{ran}(\mathcal{T}_1) \cup \mathit{ran}(\mathcal{T}_2)$. Similarly, $\mathcal{T}_1 \bullet \mathcal{T}_2$, denotes the (concatenation) transducer translating $\mathit{dom}(\mathcal{T}_1) \cdot \mathit{dom}(\mathcal{T}_2)$ into $\mathit{ran}(\mathcal{T}_1) \cdot \mathit{ran}(\mathcal{T}_2)$.

For technical reasons, we also add to the transition relation of $\mathcal{T}$ the neutral transitions $(p, \epsilon, \epsilon, 0, p)$ for each state, i.e. self-loops of weight 0, and labeled with ϵ/ϵ. Evidently, these neutral transitions do not alter any salient features of $\mathcal{T}$. However, we can now assume that any transition in the transducer $\mathcal{T}$ is always preceded by a 0-weighted transition.

Now, let's assume that all transducers have their states labeled by consecutive integers starting from 1. We denote with $\mathcal{T}_{i,j}$ the transducer obtained from $\mathcal{T}$, by shifting the set of initial states to be $\{i\}$ and the final states to be $\{j\}$.

Also, let $\mathbf{0}(\mathcal{T})_{i,j}$ be the transducer obtained from $\mathcal{T}_{i,j}$ by deleting all transitions with cost 1.

Finally, for $\{i, j\} \subset \{1, \ldots, n\}$, we consider the set of elementary transducers $\mathbf{1}_{i,j}(\mathcal{T})$, each obtained from $\mathcal{T}$ by retaining only transitions between i and j, and only those that have cost 1. Observe that, a transducer $(\mathbf{0}(\mathcal{T}))_{i,j}$ can be a full-fledged transducer i.e. it can contain loops, while an elementary transducer $\mathbf{1}_{i,j}(\mathcal{T})$ is simple in the sense that it does not contain any loops.

Given a transducer $\mathcal{T} = (\{1, \ldots, n\}, \Delta, \Omega, \tau, S, F)$ (as constructed in the previous section)[4], we wish to compute a transducer $\mathbf{k}(\mathcal{T})$, such that

$$\mathit{dom}(\mathbf{k}(\mathcal{T})) = \{w \in \mathit{dom}(\mathcal{T}) : c_{\mathcal{T}}(w, \mathit{ran}(\mathcal{T})) \leq k\}.$$

Intuitively, the $\mathit{dom}(\mathbf{k}(\mathcal{T}))$ would capture the set of words in $\mathit{dom}(\mathcal{T})$ that have a $\mathcal{T}$-cost of not more than k. Clearly, if we are able to construct $\mathbf{k}(\mathcal{T})$, then we can decide whether or not

$$c_{\mathcal{T}}(\mathit{dom}(\mathcal{T}), \mathit{ran}(\mathcal{T})) \leq k,$$

by testing the (regular) language equality $\mathit{dom}(\mathbf{k}(\mathcal{T})) = \mathit{dom}(\mathcal{T})$. Hence, by this and Theorem 2, we cast the decision of the k-boundedness into a pure regular language equivalence test, which can be done in polynomial space. Furthermore, as we show, our construction for $\mathbf{k}(\mathcal{T})$ is such that $\mathit{ran}(\mathbf{k}(\mathcal{T})) = (\mathit{ran}(\mathcal{T}))^{(k)}$, thereby giving a constructive method for obtaining $Q^{(k)}$.

We will construct $\mathbf{k}(\mathcal{T})$ by a recursive algorithm obtained from the following equations:

$$\mathbf{k}(\mathcal{T}) = \mathcal{T}^0 \cup \mathcal{T}^1 \cup \ldots \cup \mathcal{T}^k$$

where $\mathcal{T}^0$ is an elementary transducer with self-loop transitions $(i, \epsilon, \epsilon, 0, i)$, for each state i, which is both initial and final, and for $1 \leq h \leq k$

$$\mathcal{T}^h = \bigcup_{i \in S, j \in F} \mathcal{T}^h_{i,j}$$

where

$$\mathcal{T}^h_{i,j} = \begin{cases} \bigcup_{m \in \{1,\ldots,n\}} \mathcal{T}^{h/2}_{i,m} \bullet \mathcal{T}^{h/2}_{m,j} & \text{for } h \text{ even} \\ \bigcup_{m \in \{1,\ldots,n\}} \mathcal{T}^{(h-1)/2}_{i,m} \bullet \mathcal{T}^{(h+1)/2}_{m,j} & \text{for } h \text{ odd} \end{cases}$$

for $h > 1$, and

$$\mathcal{T}^1_{i,j} = \bigcup_{\{m,l\} \subset \{1,\ldots,n\}} (\mathbf{0}(\mathcal{T}))_{i,m} \bullet \mathbf{1}_{m,l}(\mathcal{T}) \bullet (\mathbf{0}(\mathcal{T}))_{l,j}\,.$$

We can now show that indeed:

Theorem 3

$$\mathrm{dom}(\mathbf{k}(\mathcal{T})) = \{w \in \mathrm{dom}(\mathcal{T}) : c_{\mathcal{T}}(w, \mathrm{ran}(\mathcal{T})) \leq k\}.$$

Moreover, we have the following theorem, which provides a constructive method for obtaining $Q^{(k)}$.

[4] In fact our construction can be applied to any weighted transducer with a slight modification of $\mathcal{T}^0$ (see below).

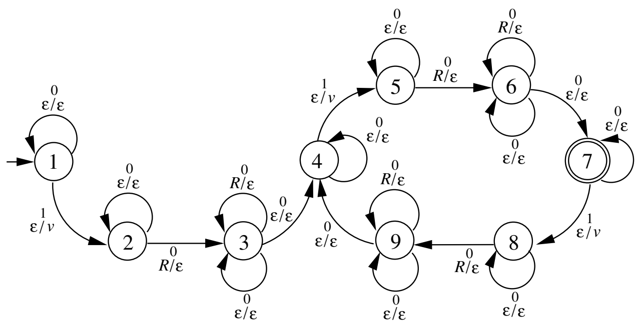

Figure 2. *Enhanced transducer*

Theorem 4 ran$(\mathbf{k}(\mathcal{T})) = Q^{(k)}$.

PROOF SKETCH. By the construction of $\mathbf{k}(\mathcal{T})$, we have captured in it all the paths of $\mathcal{T}$, which are weighted less or equal to k, and nothing else. From this, and the construction of $\mathcal{T}$, our claim follows. ■

Continuing our example of Figure 1, we show the transducer enhanced with state labels, and 0-weighted self-loops, in Figure 2.

Let $k = 2$. In the forward direction of recursion, we will reach the point where we need to construct some of the $(\mathbf{0}(\mathcal{T}))_{\_,\_}$ transducers. We observe that many of the $(\mathbf{0}(\mathcal{T}))_{i,j}$ (where $\{i, j\} \subseteq \{1, \ldots, 9\}$) are empty, such as $(\mathbf{0}(\mathcal{T}))_{1,2}$, $(\mathbf{0}(\mathcal{T}))_{3,2}$ etc. Considering those $(\mathbf{0}(\mathcal{T}))_{\_,\_}$, which are not empty, we try to compute $\mathcal{T}^1$, which results in being empty.

Next, for computing $\mathcal{T}^2$, the recursion will ask for some of the $\mathcal{T}^1_{\_,\_}$ to be computed. Of those, the non-empty ones are: $\mathcal{T}^1_{1,2}$, $\mathcal{T}^1_{1,3}$, $\mathcal{T}^1_{1,4}$, $\mathcal{T}^1_{2,7}$, $\mathcal{T}^1_{3,7}$, and $\mathcal{T}^1_{4,7}$.

As the recursion rewinds, based on the above transducers, we compute $\mathcal{T}^2$, which is the union of the following non-empty concatenations: $\mathcal{T}^1_{1,2} \bullet \mathcal{T}^1_{2,7}$, $\mathcal{T}^1_{1,3} \bullet \mathcal{T}^1_{3,7}$, $\mathcal{T}^1_{1,4} \bullet \mathcal{T}^1_{4,7}$.

It is easy to see that all the above three resulting transducers have *ran* equal to $Q^{(2)}$, which is v^2. Also, they have *dom* equal to $R^+.R^+$, that is equivalent to the *dom* of the full transducer, which is $(R^+.R^+)^+$. Thus, our Ω query $(v^2)^+$ is indeed 2-bounded and the equivalent non-recursive query can be obtained by the *ran* of the above transducers, which as mentioned above is v^2.

Notably, writing $\mathcal{T}^h_{i,j} = \bigcup_{m\in\{1,\ldots,n\}} \mathcal{T}^{h/2}_{i,m} \bullet \mathcal{T}^{h/2}_{m,j}$ (supposing h is even) instead of naively writing equivalently $\mathcal{T}^h_{i,j} = \bigcup_{m\in\{1,\ldots,n\}} \mathcal{T}^{h-1}_{i,m} \bullet \mathcal{T}^1_{m,j}$, makes us very efficient with respect to h (and in turn with respect to k) for computing $\mathcal{T}^h_{i,j}$ (and in turn $\mathcal{T}^k_{i,j}$). In order to see that, suppose for simplicity that h is a power of 2. Now, from our equation $\mathcal{T}^h_{i,j} = \bigcup_{m\in\{1,\ldots,n\}} \mathcal{T}^{h/2}_{i,m} \bullet \mathcal{T}^{h/2}_{m,j}$, we have that $\mathcal{T}^2_{i,j}$ will be a union of n transducers of size $2p$ (where p, the upper bound on the sizes of $\mathcal{T}^1_{\_,\_}$'s, is a polynomial on n), $\mathcal{T}^4_{i,j}$ will be a union of n transducers of size $4np$, $\mathcal{T}^8_{i,j}$ will be a union of n transducers of size $8n^2p$, and so on. Hence, by using our recurrence equation we will get a resulting transducer $\mathcal{T}^h_{i,j}$, which is a union of n transducers of length $hn^{log_2 h-1}p$, i.e. the size of $\mathcal{T}^h_{i,j}$ will be $hn^{log_2 h}p$. In other words, had we used the equivalent equation $\mathcal{T}^h_{i,j} = \bigcup_{m\in\{1,\ldots,n\}} \mathcal{T}^{h-1}_{i,m} \bullet \mathcal{T}^1_{m,j}$, the transducers $\mathcal{T}^h_{i,j}$ would be a union of n transducers of size pn^{h-1}, i.e. the total size would be pn^h.

We are now ready to show the following theorem.

Theorem 5 *The k-boundedness problem is in PSPACE with respect to the size of the query. Furthermore, the decision can be made in space sub-exponential with respect to k.*

PROOF. Recall that deciding k-boundedness, by Theorem 2 and Theorem 3, amounts to testing the language equality *dom*$(\mathbf{k}(\mathcal{T}))$ = *dom*$(\mathcal{T})$. Now, from the above discussion it is clear that the size of $\mathbf{k}(\mathcal{T})$ is $\mathcal{O}(kn^{log_2 k}p)$. So, we can test the language equivalence *dom*$(\mathbf{k}(\mathcal{T}))$ = *dom*$(\mathcal{T})$ in space polynomial (see [17]) in the size of $\mathcal{T}$ (which is polynomial in the size of Q), and sub-exponential on k. ■

We turn now on the lower bound for deciding the k-boundedness. Through a reduction from the universality problem for NFA's we show that

Theorem 6 *The problem of deciding k-boundedness is PSPACE-hard.*

Finally, Theorem 5 and Theorem 6 imply

Corollary 1 *The problem of k-boundedness is PSPACE-complete with respect to the size of the query.*

6 Deciding Finite Boundedness

Now, consider the weighted automaton $\mathcal{A}$, that we get if we project out the output column of the transition relation of the transducer $\mathcal{T} = (P_{\mathcal{T}}, \Delta, \Omega, \tau_{\mathcal{T}}, S, F)$, that we, in Section 4, constructed from a query Q. Formally, $\mathcal{A}$ $= (P_{\mathcal{T}}, \Delta, \tau_{\mathcal{A}}, S, F)$, where

$$\begin{aligned}\tau_{\mathcal{A}} \quad = \quad & \{(p, \epsilon, 1, q) : (p, \epsilon, v, 1, q) \in \tau_{\mathcal{T}}\} \cup \\ & \{(p, \epsilon, 0, q) : (p, \epsilon, \epsilon, 0, q) \in \tau_{\mathcal{T}}\} \cup \\ & \{(p, R, 0, q) : (p, R, \epsilon, 0, q) \in \tau_{\mathcal{T}}\}.\end{aligned}$$

Let p and q be two states of $\mathcal{A}$, and let π be a path between them, spelling a word w. Note that there can be more than one path [5] between p and q spelling w. In reasoning about the boundedness, we will be interested in the "best" path(s) spelling w, i.e. the one(s) with the smallest weight. Let therefore

$$d_{\mathcal{A}}(p,w,q) = \mathit{inf}\{weight(\pi) : \pi \text{ is a path spelling } w, \text{ from } p \text{ to } q \text{ in } \mathcal{A}\}.$$

Also, for two subsets P_1 and P_2 of states, we define

$$d_{\mathcal{A}}(P_1,w,P_2) = \mathit{inf}\{d_{\mathcal{A}}(p,w,q) : p \in P_1 \text{ and } q \in P_2\}.$$

Now, we define the *distance* of $\mathcal{A}$, as

$$d(\mathcal{A}) = \mathit{sup}\{d_{\mathcal{A}}(S,w,F)\}.$$

We say that a weighted automaton $\mathcal{A}$ is *limited* if $d(\mathcal{A}) < \infty$.

Based on the these definitions, and the construction of the weighted automaton $\mathcal{A}$, the following theorem can be shown.

Theorem 7 *Q is finitely bounded iff $\mathcal{A}$ is limited.*

Hence, the finite boundedness is reducible to the limitedness of weighted automata. Since such an automaton is constructible in polynomial time on the size of Q, we have that the reduction is polynomial as well.

Now, we show how to efficiently transform the weighted automaton $\mathcal{A}$, that we obtain from the transducer $\mathcal{T}$, into one with ϵ-free transitions, is such a way that the essential features of $\mathcal{A}$ are preserved.

From the automaton $\mathcal{A}$ we will construct another "distance equivalent" automaton $\mathcal{B}$. We shall use ϵ-*closure*(p), similarly to [16], to denote the set of all states q such that there is path π, from p to q in $\mathcal{A}$, spelling ϵ.

Obviously, we will keep all the non-ϵ transitions of $\mathcal{A}$ in the automaton $\mathcal{B}$, that we are constructing.

Now, we will insert an R-transition ($R \neq \epsilon$) in $\mathcal{B}$ from a state p to a state q whenever there is in $\mathcal{A}$ a path π, spelling ϵ, from p to an intermediate state r and there is an R-transition, from that state r to the state q. Formally, if $\mathcal{A} = (P, \Delta, \tau_{\mathcal{A}}, S, F)$, then $\mathcal{B} = (P, \Delta, \tau_{\mathcal{B}}, S, G)$, where

$$G = F \cup \{s : s \in S, \text{ and } \epsilon\textit{-closure}_{\mathcal{A}}(s) \cap F \neq \emptyset\}$$

and

$$\begin{aligned}\tau_{\mathcal{B}} \;\; = \;\; & \{(p,R,0,q) : (p,R,0,q) \in \tau_{\mathcal{A}}\} \cup \\ & \{(p,S,m,q) : \exists r \in \epsilon\textit{-closure}_{\mathcal{A}}(p), \\ & \text{ such that } (r,S,0,q) \in \tau\},\end{aligned}$$

where the weight m will be the weight of the *cheapest* path from p to r in $\mathcal{A}$ spelling ϵ.
It is easy to verify about the above constructed automaton $\mathcal{B}$ that

[5] Such paths could have some ϵ-transitions as well.

Lemma 1 *$L(\mathcal{B}) = L(\mathcal{A})$, and $d(\mathcal{B}) = d(\mathcal{A})$.*

Hence, we are now able to use Leung's algorithm [19], which is computationally the best known algorithm for solving the limitedness problem (in single exponential time), but for which the ϵ-freeness of the automata is essential.

Regarding the lower complexity bound, it can be shown that the notorious problem of finite power property (FPP) for regular languages can be reduced to our (query) finite boundedness problem. The FPP problem was posed initially by Brzozowski in 1966 during the SWAT (now FOCS) conference. It asked whether for a given regular language, say L, there exists an $m \in \mathbb{N}$, such that

$$L^* = \{\epsilon\} \cup L \cup L^2 \cup \ldots \cup L^m.$$

Now, this can be reduced to the finite boundedness problem by considering a single view $V = L$, a corresponding alphabet $\Omega = \{v\}$, and a query v^*.

The FPP problem remained open for more than 12 years, until shown to be decidable by Hashiguchi in 1978[6] (see [12]). Through combinatorial arguments, he presented a solution which works in exponential time. Another independent solution was obtained by Simon in [24]. Simon's solution also needs exponential time. Since then, there are no new results reported for the FPP problem (see for a review [23]). Thus, it seems highly unlikely that one can do better than EXPTIME for solving the finite boundedness problem.

Finally, as shown by Weber in [28], the FPP problem is PSPACE-hard, which implies that our boundedness problem is PSPACE-hard as well.

Acknowledgment. We would like to thank Daniel Kirsten for pointing to us the [28] paper.

References

[1] Abiteboul S., P. Buneman and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann Pulishers. San Francisco, CA, 1999.

[2] Bravo L., and Bertossi L. Disjunctive Deductive databases for computing certain and consistent answers from mediated data integration systems. *J. Applied Logic* 3(1): 329–367, 2005.

[3] Calvanese D., G. Giacomo, M. Lenzerini and M. Y. Vardi. Rewriting of Regular Expressions and Regular Path Queries. *Proc. PODS '99*, pp. 194–204.

[6] It is before his limitedness theorem for distance automata [13].

[4] Calvanese D., G. Giacomo, M. Lenzerini and M. Y. Vardi. Answering Regular Path Queries Using Views. *Proc. ICDE '00*, pp. 389–398.

[5] Deutsch A., Y. Katsis and Y. Papakonstantinou. Determining Source Contribution in Information Integration Systems. *Proc. PODS '05*, pp. 304–315.

[6] Flesca, S., and Greco, S. Rewriting queries using views. *IEEE Trans. Knowl. Data Eng.* 13(6): 980–995, 2001.

[7] Grahne G. and Mendelzon A. O. Tableau Techniques for Querying Information Sources through Global Schemas. *Proc. ICDT '99*, pp. 332–347.

[8] Grahne G., and A. Thomo. An Optimization Technique for Answering Regular Path Queries *Proc. WebDB '00*, pp. 99–104.

[9] Grahne G., and A. Thomo. Algebraic Rewritings for Optimizing Regular Path Queries. *Proc. ICDT '01*, pp. 303–315.

[10] Grahne, G., and Thomo, A. Query answering and containment for regular path queries under distortions. *Proc. FoIKS '04*, pp. 98–115.

[11] Jonson H., and Xiaoyan Q. DB2 information integrator V8.1: Under the Hood. *ARISE '04 http://www.scs.carleton.ca/ ∼nvillanu/Presentations/IBM.ppt*

[12] Hashiguchi K. A Decision Procedure for the Order of Regular Events. *TCS* 8: 69–72, 1979.

[13] Hashiguchi K. Limitedness Theorem on Finite Automata with Distance Functions. *J. Comp. Syst. Sci.* 24(2): 233–244, 1982.

[14] Hashiguchi K. Improved Limitedness Theorems on Finite Automata with Distance Functions. *TCS* 72(1): 27–38, 1990.

[15] Hashiguchi K. New upper bounds to the limitedness of distance automata. *TCS* 233(1-2): 19–32, 2000.

[16] Hopcroft J. E., and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley. Reading MA, 1979.

[17] Hunt H. B. III, D. J. Rosenkrantz, and T. G. Szymanski, On the Equivalence, Containment, and Covering Problems for the Regular and Context-Free Languages. *J. Comp. Syst. Sci.* 12(2): 222–268, 1976.

[18] Lenzerini M. Data Integration: A Theoretical Perspective. *Proc. PODS '02*, pp. 233–246.

[19] Leung H. Limitedness Theorem on Finite Automata with Distance Functions: An Algebraic Proof. *TCS* 81(1): 137–145, 1991.

[20] Levy A. Y., Mendelzon A. O., Sagiv Y., Srivastava D. Answering Queries Using Views. *Proc. PODS '95*, pp. 95–104.

[21] Mendelzon A. O., and P. T. Wood, Finding Regular Simple Paths in Graph Databases. *SIAM J. Comp.* 24(6): 1235–1258, 1995.

[22] Mendelzon A. O. G. A. Mihaila and T. Milo. Querying the World Wide Web. *Int. J. Dig. Lib.* 1(1): 57–67, 1997.

[23] Pin. J. E. Tropical Semirings, *in Idempotency*, J. Gunawardena (ed.) Cambridge University Press, pp. 50–69, 1998.

[24] Simon. I. Limited Subsets of a Free Monoid. *Proc. FOCS '78*, pp. 143–150.

[25] Simon. I. On Semigroups of Matrices over the Tropical Semiring. *Informatique Theorique et Applications* 28(3-4): 277–294, 1994.

[26] Ullman J. D. Information Integration Using Logical Views. *Proc. ICDT '97*, pp. 19–40.

[27] Vardi. M. Y. A Call to Regularity. *Proc. PCK50 - Principles of Computing & Knowledge, Paris C. Kanellakis Memorial Workshop '03*, pp. 11.

[28] Weber A. Distance Automata Having Large Finite Distance or Finite Ambiguity. *Mathematical Systems Theory* 26(2): 169–185, 1993.