To appear in Proceedings of **KR 2000**

Containment of Conjunctive Regular Path Queries with Inverse

Diego Calvanese[1], **Giuseppe De Giacomo**[1], **Maurizio Lenzerini**[1], **Moshe Y. Vardi**[2]

[1] Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
lastname@dis.uniroma1.it
http://www.dis.uniroma1.it/~*lastname*

[2] Department of Computer Science
Rice University, P.O. Box 1892
Houston, TX 77251-1892, U.S.A.
vardi@cs.rice.edu
http://www.cs.rice.edu/~vardi

Abstract

Reasoning on queries is a basic problem both in knowledge representation and databases. A fundamental form of reasoning on queries is checking containment, i.e., verifying whether one query yields necessarily a subset of the result of another query. Query containment is crucial in several contexts, such as query optimization, knowledge base verification, information integration, database integrity checking, and cooperative answering.

In this paper we address the problem of query containment in the context of semistructured knowledge bases, where the basic querying mechanism, namely regular path queries, asks for all pairs of objects that are connected by a path conforming to a regular expression. We consider conjunctive regular path queries with inverse, which extend regular path queries with the possibility of using both the inverse of binary relations, and conjunctions of atoms, where each atom specifies that one regular path query with inverse holds between two variables. We present a novel technique to check containment of queries in this class, based on the use of two-way finite automata. The technique shows the power of two-way automata in dealing with the inverse operator and with the variables in the queries. We also characterize the computational complexity of both the proposed algorithm and the problem.

1 INTRODUCTION

Querying is the fundamental mechanism for extracting information from a knowledge base. The basic reasoning task associated to querying is query answering, which amounts to compute the information to be returned as result of a query. However, there are other reasoning services involving queries that 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 is crucial in several contexts, such as query optimization, knowledge base verification, information integration, integrity checking, and cooperative answering.

Query optimization aims at improving the efficiency of query answering, and largely benefits from the possibility of performing various kinds of comparisons between query expressions. In particular, query containment checks are useful to recognize equivalent subexpressions, to avoid computing results already available, to recognize the possibility of using materialized views, and to use integrity constraints to speed-up query processing (Levy & Sagiv, 1995; Chaudhuri, Krishnamurthy, Potarnianos, & Shim, 1995; Widom, 1995; Adali, Candan, Papakonstantinou, & Subrahmanian, 1996; Buneman, Davidson, Hillebrand, & Suciu, 1996; Fernandez & Suciu, 1998; Milo & Suciu, 1999).

Recently, it has been shown that query containment is relevant for the task of knowledge base verification (Levy & Rousset, 1997). In (Levy & Rousset, 1998b), the problem of verifying whether a knowledge base produces the correct set of output for any set of input is solved by means of a method that exploits the ability of checking query containment.

One of the major issues in information integration (Fensel, Knoblock, Kushmerick, & Rousset, 1999) is to reformulate a query expressed over a unified domain representation in terms of the local sources. Several recent papers point out that query containment is essential for this purpose (Calvanese, De Giacomo, Lenzerini, Nardi, & Rosati, 1998; Levy, Rajaraman,

& Ordille, 1996; Knoblock & Levy, 1995; Friedman, Levy, & Millstein, 1999). Also, many information integration applications are developed on the web, where data are expressed using XML-like languages (Bray, Paoli, & Sperberg-McQueen, 1998; Calvanese, De Giacomo, & Lenzerini, 1999) and semistructured models (Buneman, 1997; Florescu, Levy, & Mendelzon, 1998a). This new scenario poses interesting challenges to both database and knowledge representation technologies, and query containment is one notable example of such challenges (Florescu, Levy, & Suciu, 1998b; Calvanese, De Giacomo, Lenzerini, & Vardi, 2000).

Besides the above described applications, query containment is crucial in integrity constraint checking (Gupta & Ullman, 1992; Fernandez, Florescu, Levy, & Suciu, 1999), cooperative answering (Motro, 1996), and in general in knowledge representation systems based on description logics and conceptual graphs, where it comes in the form of subsumption check, and is at the heart of all relevant reasoning tasks (Donini, Lenzerini, Nardi, & Schaerf, 1996; Eklund, Nagle, Nagle, & Gerholz, 1992; Donini, Lenzerini, Nardi, & Schaerf, 1998; Levy & Rousset, 1998a).

Needless to say, query containment is undecidable if we do not limit the expressive power of the query language. 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 on reasoning on queries (Chandra & Merlin, 1977). A conjunctive query is simply a conjunction of atoms, where each atom is built out from relation symbols and (existentially quantified) variables, and correspond to a single rule in non-recursive datalog.

Most of the results on query containment concern conjunctive queries and their extensions. In (Chandra & Merlin, 1977), NP-completeness has been established for conjunctive queries, in (Klug, 1988; van der Meyden, 1992), Π_2^p-completeness of containment of conjunctive queries with inequalities was proved, and in (Sagiv & Yannakakis, 1980) the case of queries with the union and difference operators was studied. For various classes of datalog queries with inequalities, decidability and undecidability results were presented in (Chaudhuri & Vardi, 1992) and (van der Meyden, 1992), respectively. Other papers consider the case of conjunctive query containment in the presence of various types of constraints (Aho, Sagiv, & Ullman, 1979; David S. Johnson, 1984; Chan, 1992; Levy & Rousset, 1996; Levy & Suciu, 1997), or in knowledge representation systems integrating datalog and description logics (Levy & Rousset, 1998a).

In this paper we address the problem of query containment in the context of a general form of knowledge bases, called semistructured knowledge bases. Our goal is to capture the essential features of knowledge bases as found in semantic networks, description logics, conceptual graphs, and in databases, both traditional and semistructured. For this purpose, we conceive a knowledge base as a 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.

In our framework, the basic querying mechanism is the one of regular path queries (Buneman, 1997; Calvanese, De Giacomo, Lenzerini, & Vardi, 1999; Abiteboul, Buneman, & Suciu, 1999), 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 knowledge base.

In this work, we consider *conjunctive regular path queries with inverse*, which extend regular path queries with the possibility of using both the inverse of binary relations, and conjunctions of atoms, where each atom specifies that one regular path query with inverse holds between two variables. Notably, several authors argue that these kinds of extensions are essential for making regular path queries useful in real settings (see for example (Buneman, 1997; Buneman et al., 1996; Milo & Suciu, 1999)). Conjunctive regular path queries have been studied in (Florescu et al., 1998b), where an EXPSPACE algorithm for query containment in this class is presented. However, no lower bound is provided for the problem, and, moreover, the method does not seem easily generalizable to take into account the inverse operator. The case with the inverse operator is implicitly addressed in (Calvanese, De Giacomo, & Lenzerini, 1998), where an 2EXPTIME algorithm is proposed for query containment. However, the framework investigated in (Calvanese et al., 1998) includes a rich set of constraints, and is not suited for a precise characterization of containment of conjunctive regular path queries with inverse.

We present a novel technique to check containment of queries in this class, based on the use of two-way finite automata. Differently from standard finite state automata, two-way automata are equipped with a head that can move back and forth on the input string. A transition of these kinds of automata indicates not only the new state, but also whether the head should move left, right, or stay in place. Our technique shows the

power of two-way automata in dealing with the inverse operator and with the variables in conjunctive queries. In particular, we describe an algorithm that checks containment of two queries by checking nonemptiness of a two-way automaton constructed from the two queries. The algorithm works in exponential space, and therefore has the same worst-case complexity as the best algorithm known for the case of conjunctive regular path queries without inverse (Florescu et al., 1998b).

We also prove an EXPSPACE lower bound for the computational complexity of the problem, thus demonstrating that our method is essentially optimal. Interestingly, the lower bound holds even if we disregard the inverse operator, and therefore provides the solution to the open problem of whether containment of conjunctive regular path queries could be done in PSPACE (Florescu et al., 1998b). Besides the specific result, our method provides the basis for using two-way automata in reasoning on complex queries, and can be adapted to more general forms of queries (e.g., unions of conjunctive queries) and reasoning tasks (e.g., query rewriting), as well as to other formalisms in Artificial Intelligence (e.g., temporal and dynamic logics with a backward modality, and action theories with converse).

2 KNOWLEDGE BASES AND QUERIES

We consider a *semistructured knowledge base* (KB) $\mathcal{K}$ 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 Σ'. A node represents an object, and an edge between nodes x and y labeled p represents the fact that the binary relation p holds for the pair (x, y). We denote an edge from x to y labeled by p with $x \xrightarrow{p} y$.

The basic querying mechanism on a KB is that of *regular path queries* (RPQs). An RPQ R is expressed as a regular expression or a finite automaton, and computes the set of pairs of nodes of the KB connected by a path that conforms to the regular language $L(R)$ defined by R. As we said in the introduction, we consider queries that extend regular path queries with both the inverse operator, and the possibility of using conjunctions and variables.

Formally, let $\Sigma = \Sigma' \cup \{p^- \mid p \in \Sigma'\}$ be the alphabet including a new symbol p^- for each p in Σ'. Intuitively, p^- denotes the inverse of the binary relation p. If $r \in \Sigma$, then we use r^- to mean the *inverse* of r, i.e., if r is p, then r^- is p^-, and if r is p^-, then r^- is p.

Regular path queries with inverse (RPQIs) are expressed by means of regular expressions or finite automata over Σ. Thus, in contrast with RPQs, RPQIs may use also the inverse p^- of p, for each $p \in \Sigma'$. When evaluated over a KB $\mathcal{K}$, an RPQI E computes the set $ans(E, \mathcal{K})$ of pairs of nodes connected by a semipath that conforms to the regular language $L(E)$ defined by E. A *semipath* in $\mathcal{K}$ from x to y is a sequence of the form $(y_1, r_1, y_2, r_2, y_3, \ldots, y_q, r_q, y_{q+1})$, where $q \geq 0$, $y_1 = x$, $y_{q+1} = y$, and for each y_i, r_i, y_{i+1}, either $y_i \xrightarrow{r_i} y_{i+1}$ or $y_{i+1} \xrightarrow{r_i^-} y_i$ is in $\mathcal{K}$. The semipath *conforms to* E if $r_1 \cdots r_q \in L(E)$. The semipath is *simple* if each y_i, for $i \in \{2, \ldots, q\}$, is a node that does not occur elsewhere in the semipath.

Finally, we add to RPQIs the possibility of using conjunctions of atoms, where each atom specifies that one regular path query with inverse holds between two variables. More precisely, if Φ is an alphabet of variables, then a *conjunctive regular path query with inverse* (CRPQI) Q is a formula of the form

$$Q(x_1, \ldots, x_n) \leftarrow y_1\, E_1\, y_2 \wedge \cdots \wedge y_{2m-1}\, E_m\, y_{2m}$$

where $x_1, \ldots, x_n, y_1, \ldots, y_{2m}$ range over a set $\{u_1, ..., u_k\}$ of variables in Φ, each x_i, called a *distinguished variable*, is one of $y_1, \ldots, y_{2m}$, and $E_1, \ldots, E_m$ are RPQIs.

The *answer set* $ans(Q, \mathcal{K})$ to a CRPQI Q over a KB $\mathcal{K} = (\mathcal{D}, \mathcal{E})$ is the set of tuples $(d_1, \ldots, d_n)$ of nodes of $\mathcal{K}$ 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 Q, and $(\sigma(y), \sigma(z)) \in ans(E, \mathcal{K})$ for every conjunct yEz in Q.

Example 1 Let us consider a KB of parental relationships. The CRPQI

$$\begin{aligned} Q(x_1, x_2) \leftarrow\ & \\ & x_1\, (\mathsf{father}^- \cdot \mathsf{father} \cup \mathsf{mother}^- \cdot \mathsf{mother})^+\, x_2 \wedge \\ & x_1\, (\mathsf{father} \cup \mathsf{mother})^* \cdot \mathsf{father}\, y \wedge \\ & x_2\, (\mathsf{father} \cup \mathsf{mother})^* \cdot \mathsf{mother}\, y \end{aligned}$$

returns all pairs of individuals (x_1, x_2) that are in the transitive closure of the sibling (including stepsibling) relation, and such that there is some individual y such that x_1 and x_2 have two descendants who are respectively the father and the mother of y. ∎

Given two CRPQIs Q_1 and Q_2, we say that Q_1 is *contained* in Q_2, written $Q_1 \subseteq Q_2$, if for every KB $\mathcal{K}$, $ans(Q_1, \mathcal{K}) \subseteq ans(Q_2, \mathcal{K})$. Obviously, $Q_1 \not\subseteq Q_2$ iff there is a *counterexample* KB to $Q_1 \subseteq Q_2$, i.e., a KB $\mathcal{K}$ with a tuple in $ans(Q_1, \mathcal{K})$ and not in $ans(Q_2, \mathcal{K})$.

Example 1 (cont.) It is possible to verify that the CRPQI

$$Q'(x_1, x_2) \leftarrow x_1\, \mathsf{father}^-\cdot\mathsf{father}\cdot\mathsf{mother}^-\cdot\mathsf{mother}\, x_2 \wedge x_1\, \mathsf{father}\cdot\mathsf{mother}^-\, x_2$$

is contained in Q. ∎

In order to characterize containment between CRPQIs, we introduce the notion of canonical KB. Let Q be the CRPQI

$$Q(x_1, \ldots, x_n) \leftarrow y_1\, E_1\, y_2 \wedge \cdots \wedge y_{2m-1}\, E_m\, y_{2m}$$

$\mathcal{K}$ be a KB, and ν be a total mapping from the variables $\{u_1, \ldots, u_k\}$ of Q to the nodes of $\mathcal{K}$. Then $\mathcal{K}$ is said to be *ν-canonical* for Q if:

- $\mathcal{K}$ constitutes of m simple semipaths, one for each conjunct of Q, which are node and edge disjoint, i.e., only start and end nodes can be shared between different semipaths.
- for $i \in \{1, \ldots, m\}$, the simple semipath associated to the conjunct $y_{2i-1}\, E_i\, y_{2i}$ connects the node $\nu(y_{2i-1})$ to the node $\nu(y_{2i})$, and conforms to E_i.

It is easy to see that, if $\mathcal{K}$ is ν-canonical for Q, then the tuple $(\nu(x_1), \ldots, \nu(x_n))$ belongs to $ans(Q, \mathcal{K})$, and therefore $ans(Q, \mathcal{K})$ is nonempty.

In the following, we assume that Q_h, for $h = \{1, 2\}$, are of the form

$$Q_h(x_1, \ldots, x_n) \leftarrow y_{h,1}\, E_{h,1}\, y_{h,2} \wedge \cdots \wedge y_{h,2m_h-1}\, E_{h,m_h}\, y_{h,2m_h}$$

i.e., Q_1 and Q_2 have the same distinguished variables $x_1, \ldots, x_n$, and the sets of non-distinguished variables of respectively Q_1 and Q_2 are disjoint. This assumption can be made without loss of generality, since we can rename variables and simulate equality between x and y by introducing $x \varepsilon y$ in the right hand side.

Let $\mathcal{K} = (\mathcal{D}, \mathcal{E})$ be a ν-canonical KB for Q_1. A total mapping μ from the variables of Q_2 to $\mathcal{D}$, is called a $(Q_2, \mathcal{K}, \nu)$-mapping if

- it maps distinguished variables of Q_2 into nodes of $\mathcal{K}$ corresponding to distinguished variables of Q_1, i.e., for each x_i, we have that $\mu(x_i) = \nu(x_i)$, and
- for all $j \in \{1, \ldots, m_2\}$, we have that $(\mu(y_{2,2j-1}), \mu(y_{2,2j})) \in ans(E_{2,j}, \mathcal{K})$.

Note that the existence of a $(Q_2, \mathcal{K}, \nu)$-mapping implies that $(\mu(x_1), \ldots, \mu(x_n)) \in ans(Q_2, \mathcal{K})$.

The following theorem provides an important characterization of containment between CRPQIs.

Theorem 2 *Let Q_1 and Q_2 be two CRPQIs. Then $Q_1 \not\subseteq Q_2$ iff there exists a KB $\mathcal{K}$ and a mapping ν from the variables of Q_1 to the nodes of $\mathcal{K}$ such that (i) $\mathcal{K}$ is ν-canonical for Q_1 and (ii) no $(Q_2, \mathcal{K}, \nu)$-mapping exists.*

Proof (sketch). For the if-part, it is easy to see that $\mathcal{K}$ is a counterexample to $Q_1 \subseteq Q_2$. For the only-if-part, it is possible to show that any counterexample can be transformed into a KB of the form stated in the theorem and that such KB is still a counterexample to $Q_1 \subseteq Q_2$. □

3 TWO-WAY AUTOMATA

A *two-way automaton* (Hopcroft & Ullman, 1979; Vardi, 1989) $A = (\Gamma, S, I, \delta, F)$ consists of an alphabet Γ, a finite set of states S, a set $I \subseteq S$ of initial states, a transition function $\delta : S \times \Sigma \to 2^{S \times \{-1,0,1\}}$, and a set $F \subseteq S$ of accepting states. Intuitively, a transition indicates both the new state of the automaton, and whether the head reading the input should move left (-1), right (1), or stay in place (0). If for all $s \in S$ and $a \in \Gamma$ we have that $\delta(s, a) \subseteq S \times \{1\}$, then the automaton is a traditional nondeterministic finite state automaton (also called *one-way automaton*).

A *configuration* of A is a pair consisting of a state and a position represented as a natural number. A *run* is a sequence of configurations. The sequence $((s_0, j_0), \ldots, (s_m, j_m))$ is a run of A on a word $w = a_0, \ldots, a_{n-1}$ in Γ^* if $s_0 \in I$, $j_0 = 0$, $j_m \leq n$, and for all $i \in \{0, \ldots, m-1\}$, we have that $0 \leq j_i < n$, and there is some $(t, k) \in \delta(s_i, a_{j_i})$ such that $s_{i+1} = t$ and $j_{i+1} = j_i + k$. This run is *accepting* if $j_m = n$ and $s_m \in F$. *A accepts w* if it has an accepting run on w. The set of words accepted by A is denoted $L(A)$.

It is well known that two-way automata define regular languages (Hopcroft & Ullman, 1979), and that, given a two-way automaton with n states, we can construct a one-way automaton with $O(2^{n \log n})$ states accepting the same language (Vardi, 1989).

We want to gain some intuition on how two-way automata capture computations of CRPQIs over KBs. To this end we show how a two-way automaton can be used for the fundamental task of verifying the nonemptiness of an RPQI over a given KB.

The basic idea that allows us to exploit automata is that we can represent special KBs by means of words. In particular, we consider KBs in which the domain $\mathcal{D}$ contains a fixed set $\mathcal{D}_0$ of nodes, and that are constituted by simple semipaths which are node and edge disjoint and such that the start and end nodes of each semipath are in $\mathcal{D}_0$. Each such KB $\mathcal{K} = (\mathcal{D}, \mathcal{E})$ is represented by a word $W_\mathcal{K}$ over the alphabet $\Sigma \cup \mathcal{D}_0 \cup \{\$\}$, which has the form

$$\$d_1 w_1 d_2 \$ d_3 w_2 d_4 \$ \cdots \$ d_{2m-1} w_m d_{2m} \$$$

where $d_1, \ldots, d_{2m}$ range over $\mathcal{D}_0$, $w_i \in \Sigma^+$, and the \$ acts as a separator. Specifically, $W_\mathcal{K}$ consists of one subword $d_{2i-1} w_i d_{2i}$, for each simple semipath in $\mathcal{K}$ from d_{2i-1} to d_{2i} conforming to w_i. Observe that we can represent the same KB by several words that differ in the direction considered for the semipaths and in the order in which the subwords corresponding to the semipaths appear.

Now, given an RPQI E, we build a two-way automaton A_E over the alphabet $\Sigma \cup \mathcal{D}_0 \cup \{\$\}$ such that A_E accepts a word $W_\mathcal{K}$ iff E has a nonempty answer on the KB $\mathcal{K}$ represented by $W_\mathcal{K}$. To do so we exploit the ability of two-way automata to:

- move on the word both forward and backward, which corresponds to traversing edges of the KB in both directions;
- "jump" from one position in the word representing a node to any other position (either preceding or succeeding) representing the same node.

These two capabilities ensure that the automaton evaluating the RPQI on the word simulates exactly the evaluation of the query on the KB.

To construct A_E, we assume that E is represented as a finite (one-way) automaton $E = (\Sigma, S, I, \delta, F)$ over the alphabet Σ. Then $A_E = (\Sigma_A, S_A, \{s_0\}, \delta_A, \{s_f\})$, where $\Sigma_A = \Sigma \cup \mathcal{D}_0 \cup \{\$\}$, $S_A = S \cup \{s_0\} \cup \{s^b \mid s \in S\} \cup S \times \mathcal{D}_0$, and δ_A is defined as follows:

1. $(s_0, 1) \in \delta_A(s_0, \ell)$, for each $\ell \in \Sigma_A$, and also $(s, 1) \in \delta_A(s_0, \ell)$ for each $s \in I$. These transitions place the head of the automaton in some randomly chosen position of the input string and set the state of the automaton to some randomly chosen initial state of E.
2. $(s^b, -1) \in \delta_A(s, \ell)$, for each $s \in S$ and $\ell \in \Sigma \cup \mathcal{D}_0$. At any point such transition makes the automaton ready to scan one step backward by placing it in "backward mode".
3. $(s_2, 1) \in \delta_A(s_1, r)$ and $(s_2, 0) \in \delta_A(s_1^b, r^-)$, for each transition $s_2 \in \delta(s_1, r)$ of E. These transitions correspond to the transitions of E which are performed forward or backward according to the current "scanning mode".
4. for each $s \in S$ and each $d \in \mathcal{D}_0$

$$\begin{array}{rcll} ((s,d),0) & \in & \delta_A(s,d) & \\ ((s,d),0) & \in & \delta_A(s^b,d) & \\ ((s,d),1) & \in & \delta_A((s,d),\ell), & \text{for each } \ell \in \Sigma_A \\ ((s,d),-1) & \in & \delta_A((s,d),\ell), & \text{for each } \ell \in \Sigma_A \\ (s,0) & \in & \delta_A((s,d),d) & \\ (s,1) & \in & \delta_A(s,d) & \end{array}$$

   Whenever the automaton reaches a symbol representing a node d (first and second clause), it may enter into "search (for d) mode" and move to any other occurrence of d in the word. Then the automaton exits search mode (second last clause) and continues its computation either forward (last clause) or backward (see item 2).
5. $(s, 1) \in \delta_A(s, \ell)$, for each $s \in F$ and each $\ell \in \Sigma_A$. These transitions move the head of the automaton to the end of the input string, when the automaton enters a final state.

Observe that the separator symbol \$ does not allow transitions except in "search mode". Its role is to force the automaton to move in the correct direction when exiting "search mode".

The following theorem characterizes the relationship between an RPQI and the corresponding two-way automaton.

Theorem 3 *Let E be an RPQI, $\mathcal{K}$ a KB over $\mathcal{D}$, and $W_\mathcal{K}$ the word representing $\mathcal{K}$. Then A_E accepts $W_\mathcal{K}$ iff $ans(E, \mathcal{K})$ is nonempty.*

4 CHECKING CONTAINMENT

We characterize both the upper bound and the lower bound of the problem of checking containment between CRPQIs.

4.1 UPPER BOUND

Let Q_h, for $h = \{1, 2\}$, be in the form

$$Q_h(x_1, \ldots, x_n) \leftarrow y_{h,1}\, E_{h,1}\, y_{h,2} \wedge \cdots \wedge y_{h,2m_h-1}\, E_{h,m_h}\, y_{h,2m_h}$$

and let $\mathcal{V}_1, \mathcal{V}_2 \subseteq \Phi$ be the sets of variables of Q_1 and Q_2 respectively. To show that Q_1 is not contained in

Q_2, we have to search for a counterexample KB. From Theorem 2, we know that it is sufficient to look for a KB $\mathcal{K}$ and a mapping ν from the variables of Q_1 to the nodes of $\mathcal{K}$, such that $\mathcal{K}$ is ν-canonical for Q_1, and such that no $(Q_2, \mathcal{K}, \nu)$-mapping exists.

To generate candidate counterexample KBs and associated ν-mappings, we first construct a one-way automaton A_1 that accepts the set of words of the form

$$\$d_1 w_1 d_2 \$ d_3 w_2 d_4 \$ \cdots \$ d_{2m_1-1} w_{m_1} d_{2m_1} \$$$

that represent a KB that is ν-canonical for Q_1 for some ν, where each d_i is an element of a fixed set of nodes $\mathcal{D}_0$. We take $\mathcal{D}_0$ to be $2^{\mathcal{V}_1}$ and we explicitly encode in a word accepted by A_1 the mapping ν from the variables of Q_1 to the nodes in $\mathcal{D}_0$. This requires to ensure that the elements of $\mathcal{D}_0$ appearing in a word accepted by A_1 constitute a partition of $\mathcal{V}_1$ into equivalence classes. To construct A_1, we define:

- A one-way automaton $A_1^{Q_1}$ that accepts all words of the form above, where for $i \in \{1, \ldots, m_1\}$, $y_{1,2i-1} \in d_{2i-1}$, $y_{1,2i} \in d_{2i}$, and w_i is a word in $L(E_i)$.
- A one-way automaton A_1^p that checks that in a word w the distinct symbols $d_i \in \mathcal{D}_0$ appearing in w constitute a partition of $\mathcal{V}_1$.
- A one-way automaton A_1^a that checks that in a word w, whenever two symbols d_{2i-1} and d_{2i} are adjacent, then $d_{2i-1} = d_{2i}$.

The number of states in $A_1^{Q_1}$ is polynomial in the size of Q_1 (although the alphabet is exponential), while the number of states in A_1^p and A_1^a is exponential in the number of variables in Q_1. A_1 is the product automaton of $A_1^{Q_1}$, A_1^p, and A_1^a, and hence is of exponential size in Q_1. We call a word W accepted by A_1 a *Q_1-word*, and use $\mathcal{K}_W$ and ν_W to denote the KB and mapping corresponding to Q_1.

By virtue of the correspondence between Q_1-words and KBs that are ν-canonical for Q_1, our method for checking whether $Q_1 \not\subseteq Q_2$ is based on searching for a Q_1-word W such that no $(Q_2, \mathcal{K}_W, \nu_W)$-mapping exists. Now, let W be a Q_1-word. To check whether there is no $(Q_2, \mathcal{K}_W, \nu_W)$-mapping, we define a two-way automaton A_3 that checks the existence of such a mapping, and then complement A_3, obtaining an automaton A_4.

In order to define A_3, we represent $(Q_2, \mathcal{K}_W, \nu_W)$-mappings as *annotations* of Q_1-words, which specify where the variables of Q_2 are being mapped to in W, and hence in $\mathcal{K}_W$. More precisely, the Q_1-word $\$\ell_1 \cdots \ell_r\$$ with each symbol $\ell_i \neq \$$ annotated with γ_i is represented by the word $\$(\ell_1, \gamma_1) \cdots (\ell_r, \gamma_r)\$$ over the alphabet $((\Sigma \cup \mathcal{D}_0) \times 2^{\mathcal{V}_2}) \cup \{\$\}$. The intended meaning is that the variables in γ_i are mapped in $\mathcal{K}_W$ to the node ℓ_i, if $\ell_i \in \mathcal{D}_0$, and are mapped to the target node of the edge corresponding to the occurrence of ℓ_i, if $\ell_i \in \Sigma$.

Given a word W' representing an annotated Q_1-word W, an automaton A_2 can check if the annotation corresponds to a $(Q_2, \mathcal{K}_W, \nu_W)$-mapping. To construct A_2, we define:

1. A one-way automaton A_2^d that checks that for every symbol $d \in \mathcal{D}_0$ containing a distinguished variable x of Q_1, every occurrence of d in W is annotated in W' with a set of variables containing x.
2. A one-way automaton A_2^s that checks that for every variable $y \in \mathcal{V}_2$, either y appears in the annotation of at most one symbol in Σ, or it appears in the annotation of every occurrence of a symbol $d \in \mathcal{D}_0$.
3. A one-way automaton $A_2^\$$ that checks that every occurrence in W of a symbol preceding a \$ is annotated in W' with the same set of variables as the symbol preceding it.
4. A two-way automaton $A_2^{Q_2}$ that checks that for all $i \in \{1, \ldots, m_2\}$, the atom $y_{2,2i-1} E_{2,i} y_{2,2i}$ of Q_2 is satisfied in $\mathcal{K}_W$, i.e., there are symbols ℓ_1, ℓ_2 in W annotated in W' with γ_1 and γ_2 respectively, such that $y_{2,2i-1} \in \gamma_1$, $y_{2,2i} \in \gamma_2$, and the pair of nodes corresponding to ℓ_1 and ℓ_2 is in $ans(E_{2,i}, \mathcal{K}_W)$. To build $A_2^{Q_2}$ we exploit the construction in Section 3.

The number of states in $A_2^{Q_2}$ is polynomial in the size of Q_2, while the number of states in A_2^d, A_2^s, and $A_2^\$$ is exponential in the number of variables in Q_2. A_2 is the product automaton of A_2^d, A_2^s, $A_2^\$$, and of the one-way automaton equivalent to $A_2^{Q_2}$ [1]. Hence A_2 is of exponential size in Q_2.

Next we define the one-way automaton A_3 that simulates the guess of an annotation of a Q_1-word, and emulates the behaviour of A_2 on the resulting annotated word. The simulation of the guess and the emulation of A_2 can be obtained simply by constructing A_2 and then projecting out the annotation from the

[1] Notice that the number of states of the one-way automaton equivalent to $A_2^{Q_2}$ does not depend on the size of the alphabet, which is exponential in the number of variables of Q_1.

transitions. The idea behind this construction is that a path in A_3 from an initial state to a final state that leads to the acceptance of a non-annotated word W, corresponds to a path in A_2 that leads to the acceptance of a word W' which represents W with some annotation, and vice-versa. Observe that A_3 has the same number of states as A_2.

Let us stress that projecting out the annotations from the transitions corresponds to guess them. However, in order for the guesses to be meaningful the automaton must be one-way, since with a two-way automaton we cannot ensure that we make the same guess each time we pass over the same position in a word.

Finally, we define the one-way automaton A_4 as the complement of A_3.

Theorem 4 *Let Q_1 and Q_2 be two CRPQIs and A_1 and A_4 be as specified above. Then $Q_1 \not\subseteq Q_2$ iff $A_1 \cap A_4$ is nonempty.*

Proof (sketch). By Theorem 2, to check $Q_1 \not\subseteq Q_2$, it is sufficient to find a KB $\mathcal{K}$ and a mapping ν from the variables of Q_1 to the nodes of $\mathcal{K}$, such that $\mathcal{K}$ is ν-canonical for Q_1, and such that no $(Q_2, \mathcal{K}, \nu)$-mapping exists. Each word W accepted by A_1 represents a KB $\mathcal{K}_W$ and a mapping ν_W such that $\mathcal{K}_W$ is ν_W-canonical for Q_1. A_4 accepts a Q_1-word W iff there is no annotation of W that represents a $(Q_2, \mathcal{K}_W, \nu_W)$-mapping. Thus, $A_1 \cap A_4$ is nonempty iff there exists a Q_1-word W such that $\mathcal{K}_W$ is canonical for Q_1, and is such that no $(Q_2, \mathcal{K}_W, \nu_W)$-mapping exists. Hence, $A_1 \cap A_4$ is nonempty iff Q_1 is not contained in Q_2. □

Theorem 5 *Given two CRPQIs Q_1 and Q_2, checking whether $Q_1 \subseteq Q_2$ can be done in EXPSPACE.*

Proof. By theorem 4, $Q_1 \subseteq Q_2$ iff $A_1 \cap A_4$ is empty. The size of A_1 is exponential in the size of Q_1, the size of A_2 is exponential in the size of Q_2, the size of A_3 is polynomial in the size of A_2, and finally, the size of A_4 is exponential in the size of A_3. Therefore, the size of A_4 is doubly exponential in the size of Q_2. However, to check whether $A_1 \cap A_4$ is empty we do not need to construct A_4 explicitly. Instead, starting from A_3, we construct A_4 "on-the-fly"; whenever the emptiness algorithm wants to move from a state s_1 of the intersection of A_1 and A_4 to a state s_2, it guesses s_2 and checks that it is directly connected to s_1. Once this has been verified, the algorithm can discard s_1. Thus, at each step the algorithm needs to keep in memory at most two states and there is no need to generate all of A_4 at any single step of the algorithm. □

4.2 LOWER BOUND

Next we show an EXPSPACE lower bound for containment of conjunctive regular path queries without inverse (CRPQs). This closes the open problem in (Florescu et al., 1998b) on whether containment of CRPQs could be done in PSPACE.

To prove the result we exploit a reduction from tiling problems (van Emde Boas, 1982, 1997; Berger, 1966). A *tile* is a unit square of one of several types and the *tiling problem* we consider is specified by means of a finite set Δ of tile types, two binary relations H and V over Δ, representing horizontal and vertical adjacency relations, respectively, and two distinguished tile types $t_S, t_F \in \Delta$. The tiling problem consists in determining whether, for a given number n in unary, a region of the integer plane of size $2^n \times k$, for some k, can be tiled consistently with the adjacency relations H and K, and with the left bottom tile of the region of type t_S and the right upper tile of type t_F. Using a reduction from acceptance of EXPSPACE Turing machines analogous to the one in (van Emde Boas, 1997), it can be shown that this tiling problem is EXPSPACE-complete.

Theorem 6 *The problem of checking whether $Q_1 \subseteq Q_2$, where Q_1 and Q_2 are two CRPQs, is EXPSPACE-hard.*

Proof (sketch). Let $T = (\Delta, H, V, t_S, t_F)$ be an instance of the EXPSPACE-complete tiling problem above and n a number in unary. The alphabet is $\Sigma = \Delta \cup \{0, 1\}$. The query Q_1 is

$$Q_1(x_1, x_2) \leftarrow x_1\, E\, x_2$$

where the regular expression in the right hand side is

$$E = 0^n \cdot t_S \cdot ((0+1)^n \cdot \Delta)^* \cdot 1^n \cdot t_F.$$

Thus, a word in $L(E)$ consists of a sequence of *blocks*, each block consists of an n-bit *address* and a tile in Δ. We intend the sequence of addresses to behave as an n-bit counter, starting with 0^n and ending with 1^n. A word in $L(E)$ encodes a tiling if each pair of adjacent tiles is consistent with H and each pair of tiles that have the same address, where there is no tile in between them with the same address, is consistent with V. Thus, a word in $L(E)$ does not encode a tiling, if it contains an *error*. An error is a pair of blocks that exhibit an incorrect behavior of the counter or that has a pair of tiles that is inconsistent with H or V. We detect errors using the query Q_2, which is

$$Q_2(x_1, x_2) \leftarrow x_1\, E_1\, y_1 \wedge \Big(\bigwedge_{i \in \{0,\ldots,n\}} y_1\, F_i\, y_2 \Big) \wedge y_2\, E_1\, x_2$$

where E_1 is the regular expression $((0+1)^n \cdot \Delta)^*$, and F_i, for $i \in \{0, \ldots, n\}$ are constructed as explained below. The intuition is that y_1 and y_2 map to a pair that represents an error.

We define a regular expression F_C that detects adjacent blocks with an error in the address bits. F_C can be constructed by encoding an n-bit counter (Börger, Gräedel, & Gurevich, 1997).

We define a regular expression F_H that detects adjacent blocks in which the tiles do not respect the horizontal adjacency relation H:

$$F_H = \sum_{(t_1,t_2) \notin H} ((0+1)^n \cdot \Delta)^* \cdot (0+1)^n \cdot t_1 \cdot (0+1)^n \cdot t_2 \cdot ((0+1)^n \cdot \Delta)^*$$

In order to construct a regular expression that detects a sequence of 2^n+1 blocks, in which the tiles in the first and last block do not respect the vertical adjacency relation V, we define:

$$G_0 = \sum_{(t_1,t_2) \notin V} (0+1)^n \cdot t_1 \cdot ((0+1)^n \cdot \Delta)^* \cdot (0+1)^n \cdot t_2$$

and, for $i \in \{1, \ldots, n\}$, we define $G_i = G_i^0 + G_i^1$, where for $b \in \{0,1\}$ ($\overline{b}$ denotes the complement of bit b):

$$\begin{aligned} G_i^b = {} & (0+1)^{i-1} \cdot b \cdot (0+1)^{n-i} \cdot \Delta \cdot \\ & (0+1)^* \cdot b \cdot (0+1)^* \cdot \Delta \cdot \\ & \overline{b}^n \cdot \Delta \cdot \\ & (0+1)^* \cdot b \cdot (0+1)^* \cdot \Delta \cdot \\ & (0+1)^{i-1} \cdot b \cdot (0+1)^{n-i} \cdot \Delta \end{aligned}$$

Assuming that the address bits are correct, a word accepted by all $G_1, \ldots, G_n$ is constituted by a sequence of blocks in which the first and the last block are exactly 2^n blocks apart. This follows from the fact that the first and the last block coincide in all the address bits, and in between there is either exactly one block with all address bits equal to 0, or exactly one block with all address bits equal to 1. The intersection of $G_0, G_1, \ldots, G_n$ detects errors due to the vertical adjacency relation V.

Hence, by defining $F_i = G_i + F_H + F_C$, for $i \in \{0, \ldots, n\}$, the query Q_2 detects all three types of errors.

If there is no tiling, then every word in $L(E)$ contains an error and Q_1 is contained in Q_2. If there is a tiling, then there is a word in $L(E)$ without an error, and this word provides a counterexample to the containment of Q_1 in Q_2. □

5 CONCLUSIONS

We have presented both upper bound and lower bound results for containment of conjunctive regular path queries with inverse. This class of queries has several features that are typical of modern query languages for knowledge and data bases. In particular, it is the largest subset of query languages for XML data (Deutsch, Fernandez, Florescu, Levy, Maier, & Suciu, 1999) for which containment has been shown decidable.

The upper bound shows that adding inverse to conjunctive regular path queries does not increase the complexity of query containment. The lower bound holds also for the case without the inverse operator, and provides the answer to the question of which is the inherent complexity of checking containment of conjunctive regular path queries.

One interesting feature of our method is to demonstrate the power of two-way automata in reasoning on complex queries. The method can also be adapted to more general forms of queries and reasoning tasks. Indeed, it is easy to extend our algorithm to the case of union of conjunctive regular path queries with inverse.

Query containment is typically the first step in addressing the more involved problems of query rewriting and query answering using views. For the case of regular path queries, such problems have been studied in (Calvanese et al., 1999, 2000). We are currently working on extending these results to more powerful query languages, such as the one considered in this paper, by exploiting the techniques based on two-way automata.

Acknowledgments

This work was supported in part by the NSF grant CCR-970006, by MURST, by ESPRIT LTR Project No. 22469 DWQ (Foundations of Data Warehouse Quality), and by the Italian Space Agency (ASI) under project "Integrazione ed Accesso a Basi di Dati Eterogenee".

References

Abiteboul, S., Buneman, P., & Suciu, D. (1999). *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, Los Altos.

Adali, S., Candan, K. S., Papakonstantinou, Y., & Subrahmanian, V. S. (1996). Query caching and optimization in distributed mediator systems. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pp. 137–148.

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

Berger, R. (1966). The undecidability of the dominoe problem. *Mem. Amer. Math. Soc.*, *66*, 1–72.

Börger, E., Gräedel, E., & Gurevich, Y. (1997). *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer-Verlag.

Bray, T., Paoli, J., & Sperberg-McQueen, C. M. (1998). 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.

Buneman, P. (1997). Semistructured data. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'97)*, pp. 117–121.

Buneman, P., Davidson, S., Hillebrand, G., & Suciu, D. (1996). A query language and optimization technique for unstructured data. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pp. 505–516.

Calvanese, D., De Giacomo, G., & Lenzerini, M. (1998). On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pp. 149–158.

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

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

Calvanese, D., De Giacomo, G., Lenzerini, M., & Vardi, M. Y. (1999). Rewriting of regular expressions and regular path queries. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'99)*, pp. 194–204.

Calvanese, D., De Giacomo, G., Lenzerini, M., & Vardi, M. Y. (2000). Answering regular path queries using views. In *Proc. of the 16th IEEE Int. Conf. on Data Engineering (ICDE 2000)*. To appear.

Chan, E. P. F. (1992). Containment and minimization of positive conjunctive queries in oodb's. In *Proc. of the 11th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'92)*, pp. 202–211.

Chandra, A. K. & Merlin, P. M. (1977). Optimal implementation of conjunctive queries in relational data bases. In *Proc. of the 9th ACM Sym. on Theory of Computing (STOC'77)*, pp. 77–90.

Chaudhuri, S., Krishnamurthy, S., Potarnianos, S., & Shim, K. (1995). Optimizing queries with materialized views. In *Proc. of the 11th IEEE Int. Conf. on Data Engineering (ICDE'95)* Taipei (Taiwan).

Chaudhuri, S. & Vardi, M. Y. (1992). On the equivalence of recursive and nonrecursive Datalog programs. In *Proc. of the 11th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'92)*, pp. 55–66.

David S. Johnson, A. C. K. (1984). Testing containment of conjunctive queries under functional and inclusion dependencies. *J. of Computer and System Sciences*, *28*(1), 167–189.

Deutsch, A., Fernandez, M., Florescu, D., Levy, A., Maier, D., & Suciu, D. (1999). Querying XML data. *IEEE Bulletin of the Technical Committee on Data Engineering*, *22*(3), 10–18.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1996). Reasoning in description logics. In Brewka, G. (Ed.), *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pp. 193–238. CSLI Publications.

Donini, F. M., Lenzerini, M., Nardi, D., & Schaerf, A. (1998). $\mathcal{AL}$-log: Integrating Datalog and description logics. *J. of Intelligent Information Systems*, *10*(3), 227–252.

Eklund, P., Nagle, T., Nagle, J., & Gerholz, L. (Eds.). (1992). *Conceptual Structures: Current Research and Practice*. Ellis Horwood.

Fensel, D., Knoblock, C., Kushmerick, N., & Rousset, M.-C. (Eds.). (1999). *IJCAI-99 Workshop on Intelligent Information Integration*. CEUR Electronic Workshop Proceedings, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-23/.

Fernandez, M., Florescu, D., Levy, A., & Suciu, D. (1999). Verifying integrity constraints on web-sites. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*.

Fernandez, M. F. & Suciu, D. (1998). Optimizing regular path expressions using graph schemas. In *Proc. of the 14th IEEE Int. Conf. on Data Engineering (ICDE'98)*, pp. 14–23.

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

Florescu, D., Levy, A., & Suciu, D. (1998b). Query containment for conjunctive queries with regular expressions. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pp. 139–148.

Friedman, M., Levy, A., & Millstein, T. (1999). Navigational plans for data integration. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI'99)*. AAAI Press/The MIT Press.

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

Hopcroft, J. E. & Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley Publ. Co., Reading, Massachussetts.

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

Knoblock, C. & Levy, A. Y. (Eds.). (1995). *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments*, No. SS-95-08 in AAAI Spring Symposium Series. AAAI Press/The MIT Press.

Levy, A. Y., Rajaraman, A., & Ordille, J. J. (1996). Query answering algorithms for information agents. In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI'96)*, pp. 40–47.

Levy, A. Y. & Rousset, M.-C. (1996). CARIN: A representation language combining Horn rules and description logics. In *Proc. of the 12th European Conf. on Artificial Intelligence (ECAI'96)*, pp. 323–327.

Levy, A. Y. & Rousset, M.-C. (1997). 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.

Levy, A. Y. & Rousset, M.-C. (1998a). Combining horn rules and description logics in CARIN. *Artificial Intelligence*, *104(1-2)*, 165–209.

Levy, A. Y. & Rousset, M.-C. (1998b). Verification of knowledge bases based on containment checking. *Artificial Intelligence*, *101(1-2)*, 227–250.

Levy, A. Y. & Sagiv, Y. (1995). Semantic query optimization in datalog programs. In *Proc. of the 14th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'95)*, pp. 163–173.

Levy, A. Y. & Suciu, D. (1997). Deciding containment for queries with complex objects. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'97)*, pp. 20–31.

Milo, T. & Suciu, D. (1999). 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*, pp. 277–295. Springer-Verlag.

Motro, A. (1996). Panorama: A database system that annotates its answers to queries with their properties. *J. of Intelligent Information Systems*, *7*(1).

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

van der Meyden, R. (1992). *The Complexity of Querying Indefinite Information*. Ph.D. thesis, Rutgers University.

van Emde Boas, P. (1982). Dominoes are forever. In *Proc. of 1st GTI Workshop*, Rheie Theoretische Informatik UGH Paderborn, pp. 75–95 Paderborn (Germany).

van Emde Boas, P. (1997). The convenience of tilings. In Sorbi, A. (Ed.), *Complexity, Logic, and Recursion Theory*, Vol. 187 of *Lecture notes in pure and applied mathematics*, pp. 331–363. Marcel Dekker Inc.

Vardi, M. Y. (1989). A note on the reduction of two-way automata to one-way automata. *Information Processing Letters*, *30*(5), 261–264.

Widom, J. (1995). Special issue on materialized views and data warehousing. *IEEE Bulletin on Data Engineering*, *18*(2).