Query Containment and Rewriting Using Views for Regular Path Queries Under Constraints

Gösta Grahne and Alex Thomo
Concordia University
{grahne, thomo}@cs.concordia.ca

ABSTRACT

In this paper we consider general path constraints for semistructured databases. Our general constraints do not suffer from the limitations of the path constraints previously studied in the literature. We investigate the containment of regular path queries under general path constraints. We show that when the path constraints and queries are expressed by words, as opposed to languages, the containment problem becomes equivalent to the word rewrite problem for a corresponding semi-Thue system. Consequently, if the corresponding semi-Thue system has an undecidable word problem, the word query containment problem will be undecidable too. Also, we show that there are word constraints, where the corresponding semi-Thue system has a decidable word rewrite problem, but the general query containment under these word constraints is undecidable. In order to overcome this, we exhibit a large, practical class of word constraints with a decidable general query containment problem.

Based on the query containment under constraints, we reason about constrained rewritings –using views– of regular path queries. We give a constructive characterization for computing optimal constrained rewritings using views.

1. INTRODUCTION

The semistructured data model [1] is now used as the foundation on which to reason about a multitude of applications, for which a strictly relational or object-oriented data model would be either inappropriate or too heavy. The data in these applications is best formalized in terms of labeled graphs, and usually such data are found in web information systems, XML data repositories, digital libraries, communication networks, and so on.

Virtually, all the query languages for semi-structured data provide the possibility for the user to query the database through regular expressions. The design of query languages using regular path expressions is based on the observation that many of the recursive queries, which arise in practice, amount to graph traversals. In essence, these queries are graph patterns, and the answers to the query are subgraphs of the database that match the given pattern [20, 12, 7, 8]. In particular, the (sub)queries expressed by regular expressions are called regular path queries.

Regular path queries have the advantage over more expressive recursive languages, such as datalog, that many decision problems become computable. However, *navigating* a database graph, in order to answer a regular path query, is very expensive. This is because a regular expression can describe arbitrarily long paths in the database, which means in turn an arbitrary number of physical accesses. Hence, the need and methodology to optimize the evaluation of regular path queries on a database graph are by now well recognized and explored [7, 15, 16]. In those papers, query optimization is approached by computing query rewritings using views. Clearly, if cached views relevant to a query are available, then a rewriting using these views offers a substantial optimization for query evaluation.

On the other hand, there are the constraints. They are facts that we know or learn about the structure of the databases, on which a query is to be evaluated. Intuition says that if we have some knowledge about the territory we are going to navigate, then we can navigate more wisely. Notably, constraints for semistructured data are investigated in [2, 5, 6, 11]. In [2], *local path constraints* are introduced. As defined there, a path constraint with respect to a node a of the database, is a pair of regular path queries (Q_1, Q_2), such that in the intended databases, the set of nodes reachable from a along paths spelling words in Q_1, is a subset of the nodes reachable from a along paths spelling words in Q_2.

An important extension of path constraints for databases having a special root node, say r, has been considered in [5, 6]. There, a path constraint holds only from nodes a that are reachable from r, by following paths labeled with words in a prefix language which is regular as well.

In all of [2, 5, 6], the *constraint implication* problem is studied. Namely, the implication problem is to test whether a new constraint follows from the ones already known. As a constraint is a query containment in [2], by solving the implication problem, the containment problem is being solved as well. However, there are limitations. The implication is

based only on the constraints holding in (from) a particular node, and so, it doesn't take into consideration other constraints holding in nearby nodes. Also, the constraints holding in a node only imply constraints holding in that same node. As a result, the methods presented in [2] can only be used to decide containment of regular path queries starting from the same node as the local constraints, and under the assumption that there are no other constraints holding in the nearby nodes, which the queries can eventually reach.

In [5, 6], for the databases having the special root node, it is shown that in general, implication of extended path constraints is undecidable, and the papers leave open the question of query containment. Finally, we can also view [11] as solving containment of regular path queries under constraints, for restricted classes of regular path queries, namely those expressible by first order logic.

In this paper we consider general semistructured databases [7, 8, 15, 16], which do not have any special root nodes. We capture (partial) knowledge about such databases, with path constraints where each constraint is a pair of regular path queries for which the containment or equality of their answer sets on the target databases is known to hold. The queries in such constraints ask for pairs of nodes connected by paths spelling words in the corresponding queries, rather than the nodes reachable by such paths from some root. Our constraints are a semantic generalization of the path constraints in [2]. With the generalization we eliminate the afore mentioned limitations associated with the constraints in [2].

We study query containment in this general setting. Query containment is considered starting from *all* the nodes of the database, not just from a special node. Then, based on the query containment, we reason about the query rewriting using views. We define *constrained rewritings* and give a characterization that enables their computation. We demonstrate that when we take constraints into account, we can always compute more useful rewritings, which use the views optimally. As mentioned before, query rewriting using views offers substantial optimization when the views are relevant to the query, and by using constraints we make more room for such relevance.

Even if the queries and the views are to be evaluated starting from all the nodes in the database, as is shown in [16], a rewriting can be efficiently used to optimize the query evaluation in the case when the query is to be computed, or the views have been computed, starting from some nodes only. Such partial query/view evaluations usually occur when dealing with regular path atoms of conjunctive regular path queries. A formal algorithm for this case is given in [16]. The algorithm is based on the intuition of "unrewriting" when reaching nodes from which a view has not been evaluated.

Query rewriting using views also has other applications apart from traditional query optimization. In the following example we illustrate how beneficial our rewritings are in a Web based scenario. The example also shows the need for the generalized path constraints used in this paper.

Let's consider the *html*-pages of the multi-site Web of Ericsson Inc. Naturally the *html*-pages have *href*-links to other pages and so we have a (finite) graph structure that we would like to query through regular expressions, instead of "endlessly" manually browsing. For the appropriate abstraction level, let there be a function mapping the *href*-links to symbols of an alphabet. For example, links for the Ericsson Canada site could be labeled with *"canada."* By browsing, the reader can easily verify that the following constraints hold from *all* the nodes of the Ericsson Web.

- *canada . products . mobile systems . mobile internet = canada . mobile internet*
- *italy . tecnologie . bluetooth . white papers = canada . technology . bluetooth . white papers*

We can continue a long way learning constraints like these. Observe that the above constraints also hold from the nodes not having at all a link labeled *"canada"* or *"italy."* In such cases, the constraints hold because the left-hand side and the right-hand side queries have both empty answer sets.

Let's take a closer look at the first constraint. It is true because at the moment there exists a shortcut link labeled "*mobile systems*" in the main page of Ericsson Canada. As a matter of fact, such shortcuts are temporary and will be replaced with something else in a couple of days. However, we would like to answer the users still asking queries like $Q = \_^*$ *. canada . mobile internet . white papers.* By considering some important long browsing paths as regular path views, e.g. $V = \_^*$ *. canada . products . mobile systems . mobile internet . white papers*, we could rewrite Q as V *. white papers* and answer the user.

Now consider the second constraint. It is clear that, if the user gives a query having a subquery asking for Canadian technology Bluetooth white papers, and the Ericsson Canada site is down for maintenance, then we could answer the user by using a query rewriting with a corresponding path view in the Italian site. Obviously, in neither case would we be able to have a rewriting if we lacked the knowledge captured in the constraints.

2. BACKGROUND

Semistructured databases. 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, we assume that we have a universe of objects D and a finite alphabet Δ, called the *database alphabet*. Objects of D will be denoted $a, b, \ldots$. Elements of Δ will be denoted $R, S, \ldots$, and words over Δ will be denoted t, u, w, x, y .... As usually, Δ^* denotes the set of all words over Δ.

A *database* DB over (D, Δ) is a pair (N, E), where $N \subseteq D$ is a set of nodes and $E \subseteq N \times \Delta \times N$ is a set of directed edges labeled with symbols from Δ.

Path queries and constraints. A *(regular) path query*[1] Q is a finite or infinite (regular) language over the alphabet Δ. Let Q be a path query, and $DB = (N, E)$ a database. Then the *answer to* Q on DB is defined as:

$$ans(Q, DB) = \{(a, b) \in N \times N : (a, R_1, c_1), (c_1, R_2, c_2) \ldots, (c_{n-1}, R_n, b) \in E, \text{ and } R_1 R_2 \cdots R_n \in Q\}$$

A query Q_1 is *contained* in a query Q_2, denoted $Q_1 \sqsubseteq Q_2$ iff $ans(Q_1, DB) \subseteq ans(Q_2, DB)$, for all DB's. We say that Q_1 is *equivalent* to Q_2 and write $Q_1 \equiv Q_2$, when $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$. It is easy to see that the above query containment coincides with the (algebraic) language containment of Q_1 and Q_2 and the query equivalence coincides with the language equality, i.e. $Q_1 \sqsubseteq Q_2$ iff $Q_1 \subseteq Q_2$ and $Q_1 \equiv Q_2$ iff $Q_1 = Q_2$.

If for two queries Q_1 and Q_2, we have in general $Q_1 \not\sqsubseteq Q_2$, we could be interested in the set $\{DB\}$ of databases, such that $ans(Q_1, DB) \subseteq ans(Q_2, DB)$. Clearly, an expression $Q_1 \sqsubseteq Q_2$ could be seen as a constraint restricting the set of databases to only those satisfying the containment. We now proceed with a the following definition:

Definition 1.

1. A *path constraint* is an expression of the form $Q_1 \sqsubseteq Q_2$, where Q_1 and Q_2 are path queries.
2. A database DB *satisfies* a path constraint, denoted $DB \models Q_1 \sqsubseteq Q_2$, if $ans(Q_1, DB) \subseteq ans(Q_2, DB)$.
3. DB satisfies a set $\mathcal{C}$ of path constraints, denoted by $DB \models \mathcal{C}$, if it satisfies each constraint in $\mathcal{C}$.
4. A query Q_1 is contained in a query Q_2 under a finite set of constraints $\mathcal{C}$, denoted $Q_1 \sqsubseteq_{\mathcal{C}} Q_2$, if for each database DB such that $DB \models \mathcal{C}$, we also have $ans(DB, Q_1) \subseteq ans(DB, Q_2)$.
5. A query Q_1 is equivalent to a query Q_2 under a finite set of constraints $\mathcal{C}$, denoted $Q_1 \equiv_{\mathcal{C}} Q_2$, if $Q_1 \sqsubseteq_{\mathcal{C}} Q_2$ and $Q_2 \sqsubseteq_{\mathcal{C}} Q_1$.
6. If two queries Q_1 and Q_2 are *words*, i.e., simply sequences of labels, the constraint $Q_1 \sqsubseteq Q_2$ is called a *word constraint*. Word constraints will also be written as $w_1 \sqsubseteq w_2$, when Q_1 equals w_1 and Q_2 equals w_2.

We can easily see now, that the query containment (equivalence) under constraints, no longer coincides with the containment (equality) of regular languages.

Rewrite systems. A (semi-Thue) *rewrite system* $\mathcal{R}$ is a finite subset of $\Delta^* \times \Delta^*$. The elements of $\mathcal{R}$ are called *rewrite rules*. A rewrite system $\mathcal{R}$ induces a *single-step reduction relation* $\rightarrow_{\mathcal{R}}$ over Δ^* defined as

$$\rightarrow_{\mathcal{R}} = \{(v, w) : v = xty \text{ and } w = xuy \text{ for some } (t, u) \in \mathcal{R}, \text{ and } x, y \in \Delta^*\}.$$

We denote with $\stackrel{*}{\rightarrow}_{\mathcal{R}}$ the reflexive and transitive closure of $\rightarrow_{\mathcal{R}}$. Testing whether a given pair (v, w) is an element of $\stackrel{*}{\rightarrow}_{\mathcal{R}}$ is called the *rewrite problem* for $\stackrel{*}{\rightarrow}_{\mathcal{R}}$. We shall sometimes use infix notation for the reduction relation and write $(u, w) \in \rightarrow_{\mathcal{R}}$ as $u \rightarrow_{\mathcal{R}} v$. For $\stackrel{*}{\rightarrow}_{\mathcal{R}}$ the convention is similar.

The following fundamental result is well known (see e.g. [4])

Theorem 1. *The rewrite problem is undecidable in general.*

We define the set of *rewrite ancestors* and *rewrite descendants* of a word w, with respect to $\mathcal{R}$, to be $anc_{\mathcal{R}}(w) = \{x \in \Delta^* : x \stackrel{*}{\rightarrow}_{\mathcal{R}} w\}$ and $desc_{\mathcal{R}}(w) = \{x \in \Delta^* : w \stackrel{*}{\rightarrow}_{\mathcal{R}} x\}$, respectively. For a language $L \subseteq \Delta^*$ we set $anc_{\mathcal{R}}(L) = \cup_{w \in L} anc_{\mathcal{R}}(w)$, and $desc_{\mathcal{R}}(L) = \cup_{w \in L} desc_{\mathcal{R}}(w)$,

We say that two Δ-words u and w are equivalent with respect to $\mathcal{R}$ iff $v \stackrel{*}{\rightarrow}_{\mathcal{R}} w$ and $w \stackrel{*}{\rightarrow}_{\mathcal{R}} v$. We denote the equivalence class of a word w with respect to $\mathcal{R}$ by $[w]_{\mathcal{R}}$, or simply by $[w]$, if $\mathcal{R}$ is evident from the context.

Let us now consider a reduction relation we would obtain by applying the rewrite rules only in the prefix of words. Formally, we define the *prefix-reduction relation*

$$\mapsto_{\mathcal{R}} = \{(v, w) : v = ty, \text{ and } w = uy, \text{ for some } (t, u) \in \mathcal{R} \text{ and } y \in \Delta^*\}.$$

We will denote the reflexive an transitive closure of $\mapsto_{\mathcal{R}}$ with $\stackrel{*}{\mapsto}_{\mathcal{R}}$.

We define the set of *prefix rewrite ancestors* and *prefix rewrite descendants* of a word w, with respect to $\mathcal{R}$, to be $panc_{\mathcal{R}}(w) = \{x \in \Delta^* : x \stackrel{*}{\mapsto}_{\mathcal{R}} w\}$ and $pdesc_{\mathcal{R}}(w) = \{x \in \Delta^* : w \stackrel{*}{\mapsto}_{\mathcal{R}} x\}$, respectively. For a language $L \subseteq \Delta^*$ we set $panc_{\mathcal{R}}(L) = \cup_{w \in L} panc_{\mathcal{R}}(w)$ and $pdesc_{\mathcal{R}}(L) = \cup_{w \in L} pdesc_{\mathcal{R}}(w)$.

Finite transducers and rational relations. A *finite transducer* $\mathcal{T} = (P, I, O, \delta, s, F)$ consists of a finite set of states P, an input alphabet I, and an output alphabet O, also a starting state s, a set of final states F, and a transition-output relation $\delta \subseteq P \times I^* \times P \times O^*$. Intuitively, for instance, $(p, v, q, w) \in \delta$ means that if the transducer is in state p and reads word v it can go to state q and emit the word w. For a given word $v \in I^*$, we say that a word $w \in O^*$ is an *output of* $\mathcal{T}$ *for* v if there exists a sequence $(s, v_1, p_1, w_1) \in \delta$, $(p_1, v_2, p_2, w_2) \in \delta$, $\ldots$, $(p_{n-1}, v_n, p_n, w_n) \in \delta$ of state transitions in $\mathcal{T}$, such that $p_n \in F$, $v = v_1 \ldots v_n$ and $w = w_1 \ldots w_n$. A finite automaton is simply a transducer without output, i.e. the tuples in the transition relation are triplets of the form (p, a, q) instead of quadruplets of the form (p, a, q, b).

We shall also use the symbol $\mathcal{T}$ to denote the set of all pairs $(v, w) \in I^* \times O^*$, where w is an output of $\mathcal{T}$ when providing

[1] A query Q could be non-regular as well. However, perhaps since very few problems would be decidable, larger classes of queries have not been considered in previous literature. We also restrict ourselves to regular path-queries, except for a technicality in Section 5.

v as input. Finally, $\mathcal{T}$ can also be seen as a mapping from languages to languages, and we write

$$\mathcal{T}(L) = \{w : (v, w) \in \mathcal{T}, \text{ for some } v \in L\}.$$

It is well known that $\mathcal{T}(L)$ is regular whenever L is.

A possibly infinite subset of $\Delta^* \times \Delta^*$ will be called a *word relation*. A word relation $\mathscr{R}$ is *rational* if there exists a transducer $\mathcal{T} = (P, \Delta, \Delta, \delta, s, F)$, such that $\mathscr{R} = \mathcal{T}$. We say that the transducer $\mathcal{T}$ *recognizes* the relation $\mathscr{R}$. It is easy to see that, if we reverse the input with the output in a transducer $\mathcal{T}$, we get a transducer recognizing the inverse $\mathscr{R}^{-1}$ of the relation $\mathscr{R}$ that $\mathcal{T}$ recognizes. We call the transducer obtained in this way from T, the *inverse transducer* and denote it with $\mathcal{T}^{-1}$.

Let $\mathscr{R}$ be a word relation. We define the powers of $\mathscr{R}$ as follows: $\mathscr{R}^0 = \emptyset, \mathscr{R}^1 = \mathscr{R}$, and

$$\mathscr{R}^{i+1} = \{(vv', ww') : (v, w) \in \mathscr{R}^i, \text{ and } (v', w') \in \mathscr{R}\}.$$

The *power closure* of $\mathscr{R}$ is defined as $\mathscr{R}^{\otimes} = \bigcup_{i \in \mathbb{N}} \mathscr{R}^i$.

It is also easy to see that class of rational relations is closed under power closure. Formally, we have

LEMMA 1. *Let $\mathscr{R}$ be a rational relation. Then $\mathscr{R}^{\otimes}$ is rational.*

PROOF. Construct a transducer $\mathcal{T} = (P, \Delta, \Delta, \delta, s, F)$ for $\mathscr{R}$. Then, the transducer $\mathcal{T}' = (P, \Delta, \Delta, \delta', s, F)$, where $\delta' = \delta \cup \{(f, \epsilon, s, \epsilon) : f \in F\}$, recognizes the relation $\mathscr{R}^{\otimes}$. □

3. QUERY CONTAINMENT UNDER CONSTRAINTS

In this section we show that for word queries and constraints, the query containment problem is equivalent to the rewrite problem for general semi-Thue systems. Although this tells us that, in general, the query containment is undecidable even for word constraints, the equivalence is important because from that we will derive a characterization for reasoning about the containment of arbitrary regular path queries. In fact, there are useful subclasses of word constraints for which the (arbitrary) query containment is decidable, and we show one such subclass in the next section.

Let $\mathcal{C} = \{t_i \sqsubseteq u_i : i \in [1, n]\}$ be a set of word constraints. Consider the corresponding semi-Thue word rewriting system $\mathcal{R}_{\mathcal{C}} = \{(t_i, u_i) : i \in [1, n]\}$. With slight abuse of notation, we shall denote the induced relations by $\rightarrow_{\mathcal{C}}$, $\overset{*}{\rightarrow}_{\mathcal{C}}$, $anc_{\mathcal{C}}(\cdot)$, and $desc_{\mathcal{C}}(\cdot)$. Now, suppose we are given two word queries w_1 and w_2, and we want to test the containment $w_1 \sqsubseteq_{\mathcal{C}} w_2$. The following theorem shows that deciding the above word query containment coincides with deciding the $w_1 \overset{*}{\rightarrow}_{\mathcal{C}} w_2$ word rewrite problem.

THEOREM 2. *$w_1 \sqsubseteq_{\mathcal{C}} w_2$ iff $w_1 \overset{*}{\rightarrow}_{\mathcal{C}} w_2$.*

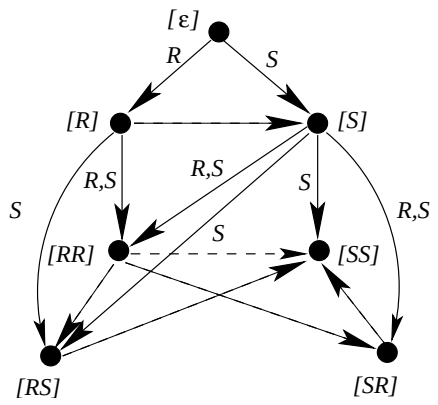

Figure 1: The construction of $DB_{\mathcal{C}}$

PROOF. It is easy to see that if $w_1 \overset{*}{\rightarrow}_{\mathcal{C}} w_2$ then $w_1 \sqsubseteq_{\mathcal{C}} w_2$. To prove the converse, we will build a database $DB_{\mathcal{C}}$ such that $DB_{\mathcal{C}} \models \mathcal{C}$ and

(§) If $DB_{\mathcal{C}} \models w_1 \sqsubseteq w_2$ then $w_1 \overset{*}{\rightarrow}_{\mathcal{C}} w_2$.

To see why this is important, suppose that we have this database $DB_{\mathcal{C}}$ and also have that $w_1 \sqsubseteq_{\mathcal{C}} w_2$. Since $w_1 \sqsubseteq_{\mathcal{C}} w_2$ and $DB_{\mathcal{C}} \models \mathcal{C}$, we get that $DB_{\mathcal{C}} \models w_1 \sqsubseteq w_2$. By (§), this means in turn that $w_1 \overset{*}{\rightarrow}_{\mathcal{C}} w_2$.

In order to construct the above mentioned database we generalize the construction of Lemma 4.4 in [2]. Since the development is neither trivial nor derivable from [2], we go in the details of the construction and its properties.

Let k be the number of symbols in the longer of the the two words w_1 and w_2. We then set $DB_{\mathcal{C}} = (N_{\mathcal{C}}, E_{\mathcal{C}})$, where

$$N_{\mathcal{C}} = \{[x] : x \in \Delta^* \text{ and } |x| \leq k\},$$

and

$$E_{\mathcal{C}} = \{([x], R, [y]) : ([x], [y]) \in N_{\mathcal{C}} \times N_{\mathcal{C}} \text{ and } y \in anc_{\mathcal{C}}(xR)\}.$$

We note here that to "really" build this database we need to be able to decide whether $y \in anc_{\mathcal{C}}(x)$ for any $[x]$ and $[y]$ in $N_{\mathcal{C}}$. Clearly, this is possible only if we are able to decide the rewrite problem for the corresponding semi-Thue system. However, what we actually show is the existence of $DB_{\mathcal{C}}$, although its construction might not be executable by a halting Turing machine.

For an example of the construction of the database $DB_{\mathcal{C}}$, let's take $\mathcal{C} = \{R \sqsubseteq S\}$, that gives a corresponding rewrite system $\{(R, S)\}$, for which the rewrite problem is decidable. The construction for $k = 2$ is presented in Figure 1. The solid arrows represent database edges, while a dashed arrow from $[x]$ to $[y]$ indicates that $y \in anc_{\mathcal{C}}(x)$.

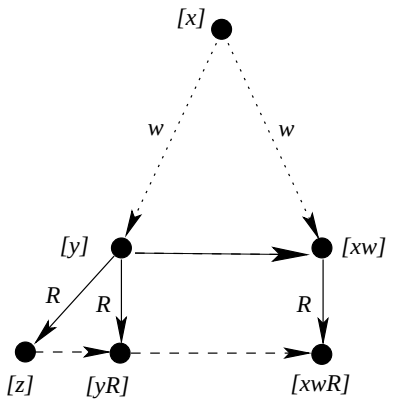

Figure 2: Proof of Lemma 2

We prove that any database $DB_{\mathcal{C}}$, constructed as described above, always satisfies the set $\mathcal{C}$ of word constraints, from which it was derived. For this we need the following lemma, which says that, starting from a node, say $[x]$, and following paths labeled with some (non-empty) word w, we reach all the nodes corresponding to the ancestors of $[xw]$. In order to simplify the notation, we will denote with $Reach([x], w)$ the set of nodes that can be reached from $[x]$ by following paths labeled with w in DB, i.e.

$$Reach([x], w) = \{[y] : ([x], [y]) \in ans(w, DB_{\mathcal{C}})\},$$

and with $Anc([x])$ we will denote the set of nodes $[y]$, such that $y \xrightarrow{*}_{\mathcal{C}} x$, i.e.

$$Anc([x]) = \{[y] \in N_c : y \in anc_{\mathcal{C}}(x)\}.$$

LEMMA 2. *For each $w \in \Delta^+$ and $[x] \in N_{\mathcal{C}}$, we have that*

$$Reach([x], w) = Anc([xw]).$$

PROOF. We proceed by induction. First, observe that by the construction, for $R \in \Delta$ we have that $[y] \in Reach([x], R)$ iff $y \in anc_{\mathcal{C}}([xR])$, i.e. $[y] \in Anc([xR])$.

Suppose (by induction on the length of w) that $Reach([x], w) = Anc([xw])$, and let $R \in \Delta$. Then, by the induction hypothesis and the construction of $DB_{\mathcal{C}}$, $Reach([x], wR)$ contains $Anc([xwR])$ [2].

Let's show the converse. If $Reach([x], wR)$ is empty, then the claimed equality follows. Let $[z]$ be in $Reach([x], wR)$. Then, there exists y, such that $[y]$ is reachable from $[x]$ with a path spelling w, and there is an R-edge from $[y]$ to $[z]$ (see Figure 2). From the induction hypothesis we get $[y] \in Anc([xw])$. Also, by the construction of $DB_{\mathcal{C}}$, $[z] \in Anc([yR])$. Now, since $[y] \in Anc([xw])$ we have that $[yR] \in Anc([xwR])$, so $Anc([yR]) \subseteq Anc([xwR])$. Thus, $[z]$ is in $Anc([xwR])$, and finally we have that $Reach([x], wR) = Anc([xwR])$. □ Lemma 2

[2]If $|xwR| > k$, then by the definition of $N_{\mathcal{C}}$, $Anc([xwR])$ is empty and the containment still holds.

Let's return to the proof of Theorem 2. For any constraint $t_i \sqsubseteq u_i$ in $\mathcal{C}$, we have $Anc([t_i]) \subseteq Anc([u_i])$. This implies that, for any node $[x] \in DB_{\mathcal{C}}$, $Anc([xt_i]) \subseteq Anc([xu_i])$, which by Lemma 2 is equivalent with $Reach([x], t_i) = Reach(\ [x], u_i)$. Since $[x]$ was an arbitrary $DB_{\mathcal{C}}$ node, we have that $DB \models t_i \sqsubseteq u_i$. Hence, $DB \models C$.

Consider now $x = \epsilon$. Lemma 2 transforms in this case into $Reach([\epsilon], w) = Anc([w])$. Then, if $DB \models w_1 \sqsubseteq w_2$, we conclude that $Reach([\epsilon], w_1) \subseteq Reach([\epsilon], w_2)$, which in turn implies that $Anc([w_1]) \subseteq Anc([w_2])$. Recall that by definition we have that $[w] \in Anc([w])$, where w is a word. So, $Anc([w_1]) \subseteq Anc([w_2])$ implies $[w_1] \in Anc([w_2])$, which finally implies $w_1 \xrightarrow{*}_{\mathcal{C}} w_2$.
□ Theorem 2

From the above theorem and Theorem 1 we get the following corollaries.

COROLLARY 1. *Query containment under constraints is undecidable.*

COROLLARY 2. *A subclass of word constraints has decidable word query containment problem iff the corresponding class of semi-Thue rewrite systems has decidable word rewrite problem.*

The second corollary is a positive result, so we would be interested in knowing whether the general query containment is decidable for a subclass of word constraints with decidable word query containment. Unfortunately, the answer to this question is negative as we show by the following lemmas and theorems. Nevertheless, they provide the basis for showing (in the next section) that a useful subclass of word constraints has decidable general query containment problem.

The following lemma is a generalization of Lemma 4.6 in [2].

LEMMA 3. *Let $\mathcal{C}$ be a finite set of word constraints and Q_1, Q_2 regular path queries. If $Q_1 \sqsubseteq_{\mathcal{C}} Q_2$ then for each $w_1 \in Q_1$ there exists $w_2 \in Q_2$ such that $w_1 \sqsubseteq_{\mathcal{C}} w_2$.*

PROOF. Let $w_1 \in Q_1$. Then, if $\mathcal{C} \models Q_1 \sqsubseteq Q_2$ we have that $\mathcal{C} \models w_1 \sqsubseteq Q_2$. Now, let's consider the database $DB_{\mathcal{C}}$ described in the proof of Theorem 2, for $k > |w_1|$. Since $DB_{\mathcal{C}}$ satisfies $\mathcal{C}$, it must also satisfy $w_1 \sqsubseteq Q_2$. It follows that

$$Reach([\epsilon], w_1) \subseteq \bigcup_{w_2 \in Q_2'} Reach([\epsilon], w_2),$$

where Q_2' is the subset of Q_2 such that for each $w \in Q_2'$, $[w]$ is a node in $DB_{\mathcal{C}}$. Now, observe that by the construction of $DB_{\mathcal{C}}$, for any two nodes $[x]$, $[y]$ we have that $[x] \in Reach([\epsilon], y)$ iff $x \in anc_{\mathcal{C}}(y)$. Since by the construction $[w_1] \in Reach([\epsilon], w_1)$, it follows that there exists $w_2 \in Q_2'$,

such that $[w_1] \in Reach([\epsilon], w_2)$. From this we have that $w_1 \in anc_{\mathcal{C}}(w_2)$. The last fact implies that $w_1 \xrightarrow{*}_{\mathcal{C}} w_2$, which as showed in Theorem 2, coincides with $w_1 \sqsubseteq_{\mathcal{C}} w_2$. □

From the above lemma and Theorem 2, we can easily see that the following is true.

THEOREM 3. *Given a finite set $\mathcal{C}$ of word constraints, and (general) regular path queries Q_1 and Q_2, we have that $Q_1 \sqsubseteq_{\mathcal{C}} Q_2$ iff $Q_1 \subseteq anc_{\mathcal{C}}(Q_2)$.*

Let $\mathcal{R}$ be a word rewriting system and $\mathcal{R}^{-1}$ its inverse, obtained by reversing the direction of the pairs in $\mathcal{R}$. Clearly, the inverse of a word rewriting system is also a word rewriting system. It is easy to verify the following lemma.

LEMMA 4. $\xrightarrow{*}_{\mathcal{R}^{-1}} = (\xrightarrow{*}_{\mathcal{R}})^{-1}$ *and* $anc_{\mathcal{R}}(L) = desc_{\mathcal{R}^{-1}}(L)$, *for any language* $L \subseteq \Delta^*$.

Now, we are ready to show that the containment for (general) regular path queries, under finite sets of word constraints with decidable word query containment, is undecidable.

THEOREM 4. *There exists a subclass of word constraints with decidable word query containment, but with undecidable query containment in general.*

PROOF. The proof is based on the notion of monadic word rewriting systems. A word rewriting system $\mathcal{R}$ is *length-reducing* if $|t| > |u|$ for each pair $(t, u) \in \mathcal{R}$. A word rewriting system $\mathcal{R}$ is *monadic* if it is length-reducing, and $u \in \Delta$ for each pair $(t, u) \in \mathcal{R}$. It is well known that the rewrite problem for length-reducing systems is decidable, and so it is for monadic systems [4].

Let us now consider the universality problem for the context free class CFG of grammars. Formally, this problem says that the language

$$\{<G> : G \in CFG \text{ and } L(G) = \Delta^*\},$$

where $<G>$ is a grammar encoding, is undecidable [25]. Let $CFG_{\geq 2}$ be the subclass of context free grammars whose productions have right side of length greater or equal to 2. Without loss of generality, we have that the slightly modified language

$$\{<G> : G \in CFG_{\geq 2} \text{ and } L(G) = \Delta^+ - \Delta\}$$

is also undecidable. We will now present a reduction from this undecidable problem.

Let G be a grammar in $CFG_{\geq 2}$ with nonterminal symbols Γ, start symbol S, and productions of the form $(head, body)$. Clearly,

$$\mathcal{R}_G = \{(body, head) : (head, body) \text{ is a production in } G\}$$

is a monadic rewrite system over the extended alphabet $\Delta \cup \Gamma$. In this proof we will consider $\Delta \cup \Gamma$ as the database alphabet.

We take $Q_1 = \Delta^*$, $Q_2 = S$, and $\mathcal{C} = \{body \sqsubseteq head : (body, head) \in \mathcal{R}_G\}$. Since $\mathcal{R}_G$ belongs to the class of monadic rewrite systems, which has a decidable word rewrite problem, from Corollary 2 it follows that $\mathcal{C}$ belongs in a subclass of word constraints with a decidable word query containment problem. Clearly, $L(G) \subseteq (\Delta^+ - \Delta)$. On the other hand, based on Lemma 4, we have that $(\Delta^+ - \Delta) \subseteq L(G)$ iff $(\Delta^+ - \Delta) \subseteq anc_{\mathcal{C}}(S) \cap (\Delta^+ - \Delta)$, which is equivalent with $(\Delta^+ - \Delta) \subseteq anc_{\mathcal{C}}(S)$, and this is finally equivalent by Theorem 3 with $Q_1 \sqsubseteq_{\mathcal{C}} Q_2$.

□

4. A SUBCLASS WITH DECIDABLE QUERY CONTAINMENT

Often, in a Web based scenario we can encounter sets of constraints, such that any left-hand side overlaps (if it does) with some right-hand side only by prefix. For example, the constraints presented in Section 1 were such ones. The high frequency of this type of constraints is because they, after a short prefix, start expressing local information about a node, say a, and the prefix is nothing else but the link (or sequence of few links) that we need to navigate from another node in order to reach a. In our example of Section 1, from any *html* page of the Ericsson Web, we can jump to the Ericsson Canada main page, only if there is a link labeled *"canada"* –the prefix– and then, various local facts can hold starting from there.

The local constraints –connected with particular nodes– have been very well motivated in [2]. However, as explained in Section 1, there are limitations associated with the way the constraints are treated in [2], most important of which are: (a) the non-extensibility of the methods to decide query containment when the queries start from other nodes than the constraints, and (b) not taking into consideration the interaction with other local constraints in nearby nodes. By prefixing the local constraints with the link label(s) needed to reach the relevant node, the local constraints are transformed into global ones, and we will give decision procedures that do not have the mentioned limitations [3].

We also relax the notion of "overlapping" to be more "generous," in the sense that it allows a left-hand side to not overlap at all with any right-hand side, and it also allows strictly internal sub-words to overlap. This relaxation of the overlapping allows for expressing not only local constraints, but also "pure" global constraints as it is the second constraint in Section 1 or the constraint *canada . events . bluetooth . latest* $\sqsubseteq$ *bluetooth . events . latest*, in which the left-hand side overlaps with the right-hand side by a strictly internal subword.

Now let's formally define our subclass of word constraints. We will start by considering rewrite systems $\mathcal{R}$. We first set $left(\mathcal{R}) = \{t : (t, u) \in \mathcal{R}, \text{ for some } u \in \Delta^*\}$, and

[3]If from some node there is no link(s) leading to the relevant node, then the constraints still hold since the sets of the reachable nodes with the left- and right-hand sides (words) are empty.

$right(\mathcal{R}) = \{u : (t,u) \in \mathcal{R}, \text{ for some } t \in \Delta^*\}$. Now we have

DEFINITION 2. ([24]). Let $\mathcal{R}$ be a rewrite system. Then $\mathcal{R}$ is said to be

1. *Internal overlapping*, if some $t \in left(\mathcal{R})$ is a substring of a word $u \in right(\mathcal{R})$ or vice versa.
2. *Right overlapping*, if there are $x, y, w \in \Delta^*$, $t \in left(\mathcal{R})$, and $u \in right(\mathcal{R})$, such that $t = xw$ and $u = wy$, for $w \neq \epsilon$.
3. *Left overlapping*, if there are $x, y, w \in \Delta^*$, $t \in left(\mathcal{R})$, and $u \in right(\mathcal{R})$, such that $u = yw$ and $t = wx$, for $w \neq \epsilon$.

If a rewrite system $\mathcal{R}$ is neither internal, nor left or right overlapping, we say that $\mathcal{R}$ is *prefix overlapping*. Clearly, a prefix overlapping system allows in addition for non-overlapping at all, or overlappings of left- with right-hand sides by strictly internal subwords.

The following result has recently been obtained by Caucal.

THEOREM 5. ([10]). *Let $\mathcal{R}$ be a prefix overlapping rewrite system. Then, we have that*

$$\overset{*}{\rightarrow}_{\mathcal{R}} = (\overset{*}{\mapsto}_{\mathcal{R}})^{\otimes}.$$

Abiteboul and Vianu have an important related recent result.

THEOREM 6. ([2]). *Let $\mathcal{R}$ be an arbitrary rewrite system and $L \subseteq \Delta^*$ a regular language. Then, the set of prefix rewrite ancestors $panc_{\mathcal{R}}(L)$ is regular as well, and computable in PTIME from an automaton for L.*

Based on the above theorem and reasoning similarly as for Lemma 4, we also have

THEOREM 7. *Let $\mathcal{R}$ be an arbitrary rewrite system and $L \subseteq \Delta^*$ a regular language. Then, the set of prefix rewrite descendants $pdesc_{\mathcal{R}}(L)$ is regular as well, and computable in PTIME from an automaton for L.*

Here we strengthen these results for $\overset{*}{\mapsto}_{\mathcal{R}}$. We show that $\overset{*}{\mapsto}_{\mathcal{R}}$ is rational for any arbitrary rewrite system $\mathcal{R}$.

We do this in order to compute $anc_{\mathcal{R}}(L)$ for a regular language L and a prefix overlapping rewrite system $\mathcal{R}$. It can be easily verified that, $anc_{\mathcal{R}}(L)$ is not equal in general with $(panc_{\mathcal{R}}(L))^{\otimes}$, as we could think at the first glance on Theorem 5.

Let $\mathcal{R}$ be a rewrite system, and (v,w) a pair of words. Then the *prefix lineage through (v,w) induced by $\mathcal{R}$* is defined as

$$plin_{\mathcal{R}}(v,w) = \{(x,y) : x \in panc_{\mathcal{R}}(v) \text{ and } y \in pdesc_{\mathcal{R}}(w)\}.$$

We have the following theorem.

THEOREM 8. *Let $\mathcal{S} = \cup\{plin_{\mathcal{R}}(t,u) : (t,u) \in \mathcal{R}\} \cup \{\epsilon,\epsilon\}$. Then, $\overset{*}{\mapsto}_{\mathcal{R}} = \mapsto_{\mathcal{S}}$.*

PROOF. We will prove that, for the rewrite system $\overset{*}{\mapsto}_{\mathcal{R}}$, $w_1 \overset{*}{\mapsto}_{\mathcal{R}} w_2$ if and only if $w_1 = w_2$, or, $w_1 = xz$ and $w_2 = yz$, where $\{x,y,z\} \subset \Delta^*$, and there exists a a pair $(t,u) \in \mathcal{R}$, such that such that $x \overset{*}{\mapsto}_{\mathcal{R}} t$ and $u \overset{*}{\mapsto}_{\mathcal{R}} y$. Clearly from this, the theorem follows.

The *If*-direction is straightforward. For the converse we will use induction on n, for $w_1 \overset{n}{\mapsto}_{\mathcal{R}} w_2$. For $n = 0$ we must indeed have $w_1 = w_2$.

Suppose that the claim is true for n, and let's show it for $n+1$. Let $w_1 \overset{n+1}{\mapsto}_{\mathcal{R}} w_2$. There exists a word v, such that $w_1 \overset{n}{\mapsto}_{\mathcal{R}} v \mapsto_{\mathcal{R}} w_2$. By the induction hypothesis $w_1 = x_1 z_1$ and $v = y_1 z_1$, and for some $(t_1, u_1) \in \mathcal{R}$ we have that $x_1 \overset{*}{\mapsto}_{\mathcal{R}} t_1$ and $u_1 \overset{*}{\mapsto}_{\mathcal{R}} y_1$.

Since $v \mapsto_{\mathcal{R}} w_2$, there exists a pair $(t_2, u_2) \in \mathcal{R}$, such that $v = t_2 z_2$ and $w_2 = u_2 z_2$. So, $v = t_2 z_2 = y_1 z_1$ and we have that either y_1 is a prefix of t_2, or vice versa.

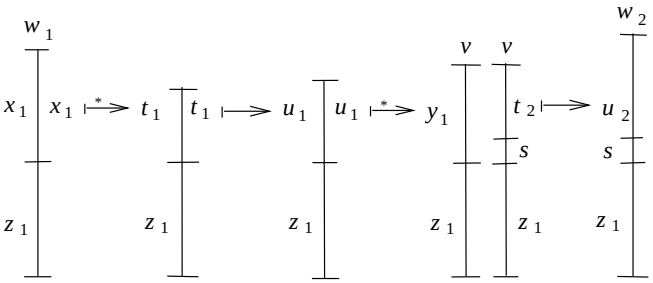

Figure 3: First case of Theorem 8

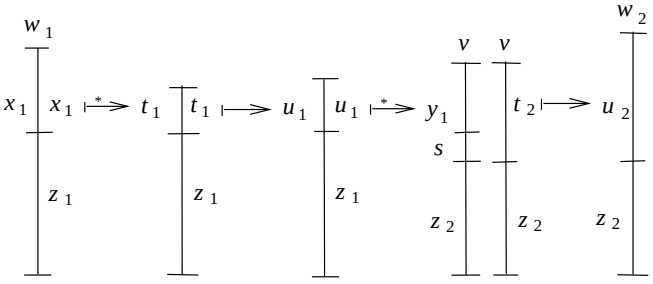

Figure 4: Second case of Theorem 8

In the first case (see Figure 3) t_2 is a prefix of y_1. So, $y_1 = t_2 s$ and $z_2 = s z_1$. Thus, $w_1 = x_1 z_1$ and $w_2 = u_2 z_2 = u_2 s z_1$. By the induction hypothesis ($x_1 \overset{*}{\mapsto}_{\mathcal{R}} t_1$ and $u_1 \overset{*}{\mapsto}_{\mathcal{R}} y_1$), we have that $x_1 \overset{*}{\mapsto}_{\mathcal{R}} t_1 \mapsto_{\mathcal{R}} u_1 \overset{*}{\mapsto}_{\mathcal{R}} y_1$. Since $y_1 = t_2 s$, $u_1 \overset{*}{\mapsto}_{\mathcal{R}} t_2 s \mapsto_{\mathcal{R}}$

$u_2 s$. On the other hand, recall that $w_2 = u_2 s z_1$. So, in total we have $w_1 = x_1 z_1$ and $w_2 = u_2 s z_1$, where $x_1 \overset{*}{\mapsto}_{\mathcal{R}} t_1$ and $u_1 \overset{*}{\mapsto}_{\mathcal{R}} u_2 s$. Hence, in this case we take $x = x_1$, $y = u_2 s$, $z = z_1$, and $(t, u) = (t_1, u_1)$.

In the second case (see Figure 4) we have $t_2 = y_1 s$ and $z_1 = s z_2$. Thus, $w_1 = x_1 z_1 = x_1 s z_2$, and by the induction hypothesis ($x_1 \overset{*}{\mapsto}_{\mathcal{R}} t_1$ and $u_1 \overset{*}{\mapsto}_{\mathcal{R}} y_1$), we have that $x_1 s \overset{*}{\mapsto}_{\mathcal{R}} t_1 s \mapsto_{\mathcal{R}} u_1 s \overset{*}{\mapsto}_{\mathcal{R}} y_1 s$. Since $y_1 s = t_2$, we have that $x_1 s \overset{*}{\mapsto}_{\mathcal{R}} t_2$ (recall $w_1 = x_1 s z_2$). On the other hand, recall that $w_2 = u_2 z_2$. Hence, in this case we take $x = x_1 s$, $y = u_2$, $z = z_2$, and $(t, u) = (t_2, u_2)$. □

THEOREM 9. *$\overset{*}{\mapsto}_{\mathcal{R}}$ is a rational relation.*

PROOF. Observe that for a pair (v, w), $plin_{\mathcal{R}}(v, w) = panc_{\mathcal{R}}(v) \times pdesc_{\mathcal{R}}(w)$. Since from Theorem 6 and Theorem 7, $panc_{\mathcal{R}}(v)$ and $pdesc_{\mathcal{R}}(w)$ are regular languages, we have from Theorem 8 that the corresponding rewrite system $\mathcal{S}$ is a union of Cartesian products of regular languages.

It is not difficult to construct a transducer for a Cartesian product $L \times M$ of two regular languages. For this, let $\mathcal{A}_L = (P_L, \Delta, \delta_L, s_L, F_L)$ and $\mathcal{A}_M = (P_M, \Delta, \delta_M, s_M, F_M)$ be finite automata recognizing L and M, respectively. Then construct the transducer $\mathcal{T}_{L\times M} = (P_L \cup P_M, \Delta, \Delta, \delta, s_L, F_M)$, where δ contains all the tuples (p, a, q) of δ_L expanded as (p, a, q, ϵ), all the tuples (p, b, q) of δ_M expanded as (p, ϵ, q, b), the set $\{(f, \epsilon, s_M, \epsilon) : f \in F_L\}$, and nothing else. It is easily seen that $\mathcal{T}_{L\times M}$ recognizes exactly $L \times M$.

Finally, we construct a transducer for $\mapsto_{\mathcal{S}}$ by taking the (finite) union of the transducers recognizing the above Cartesian products, concatenating at the end with a transducer "leaving everything unchanged." □

From the above theorem and Theorem 5 we have the following corollary.

COROLLARY 3. *Let $\mathcal{R}$ be prefix overlapping. Then $\overset{*}{\rightarrow}_{\mathcal{R}}$ is rational.*

We say that a finite set $\mathcal{C}$ of word constraints is prefix overlapping, if the corresponding rewrite system $\mathcal{R}_{\mathcal{C}}$ is prefix overlapping. We now finally have

THEOREM 10. *Let $\mathcal{C}$ be a prefix overlapping finite set of word constraints, and Q_1 and Q_2 regular path queries. Then, $Q_1 \sqsubseteq_{\mathcal{C}} Q_2$ is decidable and complete in PSPACE.*

PROOF. From all the above it follows that the relation $\overset{*}{\rightarrow}_{\mathcal{C}}$ is rational and a transducer $\mathcal{T}_{\mathcal{C}}$ recognizing it can be computed in polynomial time. It is easy now to verify that

$$anc_{\mathcal{C}}(Q) = \mathcal{T}_{\mathcal{C}}^{-1}(Q).$$

Based on this fact and Theorem 3, we conclude that deciding the containment $Q_1 \sqsubseteq_{\mathcal{C}} Q_2$, for general path queries, under a prefix overlapping set $\mathcal{C}$ of word constraints, is equivalent with deciding $Q_1 \subseteq \mathcal{T}_{\mathcal{C}}^{-1}(Q_2)$, and this is in PSPACE. The lower bound is the same because in the absence of constraints, the query containment coincides with the algebraic containment of regular languages, which is PSPACE complete. □

5. QUERY REWRITING USING VIEWS UNDER CONSTRAINTS

We will now reason about rewriting of regular path queries using views based on the query containment under constraints. We first recall some concepts regarding rewriting regular path queries (without constraints).

Let $\mathbf{V} = \{V_1, \ldots, V_n\}$ be a set of *view definitions* with each V_i being a finite or infinite regular language over Δ. Associated with each view definition V_i there is a view name v_i. We call the set $\Omega = \{v_1, \ldots, v_n\}$ the *outer* or *view alphabet*. For each $v_i \in \Omega$, we set $def(v_i) = V_i$. The substitution *def* associates with each view name v_i in Ω alphabet the language V_i. We also extend the substitution *def* to the Δ alphabet to be the mapping associating each symbol with itself. The substitution *def* is applied to words, languages, and regular expressions in the usual way (see e. g. [17]).

Given a database DB, which is a graph where the edges are labelled with database symbols from Δ, we define the *view graph* $\mathbf{V}_{DB}$ to be the graph induced by the set

$$\bigcup_{i\in\{1,\ldots,n\}} \{(a, v_i, b) : (a, b) \in ans(V_i, DB)\}.$$

of Ω-labelled edges.

Suppose now that we want to answer a query on the view graph instead of on the database. To this end, we say that a query $Q' \subseteq \Omega^*$ is a *rewriting* of a query $Q \subseteq \Delta^*$, if $def(Q') \subseteq Q$. If $def(Q') = Q$, the rewriting Q' is said to be *exact*.

In the seminal paper [7], Calvanese et al. give an automata theoretic method for constructing the *maximally contained rewriting* $MCR_{\mathbf{V}}$, which is the set of *all* words w on Ω, such that $def(w) \subseteq Q$.

For example, let $Q = R^*S$, $V_1 = R^*$, and $V_2 = S$ [4]. Then, $MCR_{\mathbf{V}}(Q) = v_1 v_2$. Additionally, it happens to be exact.

As it can easily be verified, when the rewriting $MCR_{\mathbf{V}}$ is exact, it can be used to obtain all the answers to the query.

THEOREM 11. *For all databases DB, we have that*

$$ans(MCR_{\mathbf{V}}(Q), \mathbf{V}_{DB}) \subseteq ans(Q, DB).$$

If $MCR_{\mathbf{V}}(Q)$ is exact, then

$$ans(MCR_{\mathbf{V}}(Q), \mathbf{V}_{DB}) = ans(Q, DB).$$

[4]For simplicity, we blur the distinction between the regular languages and the corresponding regular expressions.

Unfortunately, rewritings in Ω^* can happen to not exist. Furthermore, even if there are rewritings in Ω^*, it can happen that an exact one does not exists. Suppose for example, that $Q = R_1 \cdots R_{200}$, and that we have two views, V_1 and V_2, where $V_1 = R_1 \cdots R_{99}$, and $V_2 = R_{101} \cdots R_{199}$. Then, there is no rewriting in Ω^*, and we cannot obtain any answer for Q from the view graph only.

However, in [7, 15, 16], partial rewritings have been introduced in order to deal with such situations. As discussed in [16], a partial rewriting can be used to compute the answer to the query by evaluating a partial rewriting on the view graph and, only when necessary, by consulting the database. In our example we could have a partial rewriting $Q' = V_1 R_{100} V_2 R_{200}$. A hybrid algorithm *eval* is given in [15, 16], and for this algorithm, with inputs: 1) an "exact" partial rewriting Q', 2) a database DB and 3) a view graph $\mathbf{V}_{DB}$, we have

THEOREM 12. ([15, 16]) $eval(Q', \mathbf{V}_{DB}, DB) = ans(Q, DB)$.

Formally, a language Q' over the mixed alphabet $\Omega \cup \Delta$, is said to be a partial rewriting of Q using $\mathbf{V}$, if $def(Q') \subseteq Q$. The partial rewriting is exact if $def(Q') = Q$. Since obviously Q itself is an exact partial rewriting of Q, we need a way to compare partial rewritings of the same query. To this end, a partial order $\leq^Q_{\mathbf{V}}$ is introduced in [16], where $Q_1 \leq^Q_{\mathbf{V}} Q_2$ intuitively means that the words in Q_2 have more view symbols than the words in Q_1. For instance, if we let $Q = R_1 \cdots R_{200}$, and $\mathbf{V} = \{V_1, V_2\}$ as above, we have $R_1 \cdots R_{200} \leq^Q_{\mathbf{V}} v_1 R_{100} \cdots R_{200} \leq^Q_{\mathbf{V}} v_1 R_{100} v_2 R_{200}$.

In [16] it is shown that if we set $MPR_{\mathbf{V}}(Q)$ to be the union of all $\leq^Q_{\mathbf{V}}$-maximal partial rewritings of Q, then we get an exact rewriting, in other words, $def(MPR_{\mathbf{V}}(Q)) = Q$. Also, as discussed in the mentioned paper, $MPR_{\mathbf{V}}(Q)$ is the "best in class" compared to other partial rewritings in [7, 15]. Furthermore, [16] gives an automata theoretic construction of $MPR_{\mathbf{V}}(Q)$ and shows that it can be effectively computed.

Suppose now that we have a set $\mathcal{C}$ of constraints available. The question is if can we use the constraints to get "bigger and better" rewritings. For example, let $Q = R_1 \cdots R_{99} R_{100} \cdots R_{200}$, and suppose that we have two views, V_1 and V_2, where $V_1 = S_1 \cdots S_{99}$, $V_2 = R_{101} \cdots R_{199}$, and $\mathcal{C} = \{R_i \sqsubseteq S_i : i \in [1, 99]\} \cup \{S_i \sqsubseteq R_i : i \in [1, 99]\}$. Then, $MPR_{\mathbf{V}}(Q) = R_1 \cdots R_{99} R_{100} v_2 R_{200}$. On the other hand, the rewriting $v_1 R_{100} v_2 R_{200}$ is "bigger and better" because $v_1 R_{100} v_2 R_{200}$ is $\mathcal{C}$-equivalent to Q.

In the rest of this section we show how to obtain such "biggest and best" rewritings, using constraints.

Let $\mathcal{C}$ be a finite set of constraints, and let $L \subseteq \Delta^*$. Then we define the *maximization of* L *under* $\mathcal{C}$, denoted $max_{\mathcal{C}}(L)$ to be the $\subseteq$-largest language M, such that $M \equiv_{\mathcal{C}} L$.

Intuitively, under constraints we can replace more subwords and still have a rewriting. We capture this intuition by enlarging the view languages V_i to $max_{\mathcal{C}}(V_i)$, for $i \in [1, n]$.

We then define the substitution $def_{\mathcal{C}} : \Omega \cup \Delta \rightarrow \Delta^*$ as $def_{\mathcal{C}}(v_i) = max_{\mathcal{C}}(V_i)$, for $v_i \in \Omega$, and $def_{\mathcal{C}}(R) = \{R\}$ for $R \in \Delta$.

A *$\mathcal{C}$-constrained partial $\mathbf{V}$-rewriting* of Q is a language Q' on $\Omega \cup \Delta$, such that $def_{\mathcal{C}}(Q') \sqsubseteq_{\mathcal{C}} Q$. The rewriting is said to be *$\mathcal{C}$-constrained exact* if $def_{\mathcal{C}}(Q') \equiv_{\mathcal{C}} Q$.

In order to compare different rewritings, we generalize the partial order in [16] to take constraints into account. With this partial order we want to capture the intuition that the more subwords on the Δ-alphabet that have been replaced by Ω symbols in a rewriting, the "bigger and better" the rewriting is.

Let Q_1 and Q_2 be $\mathcal{C}$-constrained $\mathbf{V}$-rewritings (over $\Omega \cup \Delta$) of Q. Then, Q_1 is "smaller" than Q_2, denoted $Q_1 \leq^Q_{\mathbf{V},\mathcal{C}} Q_2$, if it is possible to substitute by $v_{i_1}, \ldots, v_{i_k}$ some (not necessarily all) occurrences of words in $max_{\mathcal{C}}(V_{i_1})$, ..., $max_{\mathcal{C}}(V_{i_k})$ respectively, occurring as subwords in Q_1, and obtain Q_2 as a result. Also, a word in Q_1 can participate in the creation of more than one word in Q_2.

Obviously, $\leq^Q_{\mathbf{V},\mathcal{C}}$ is transitive and reflexive. Is is not antisymmetric, as for instance $\{vRR, vv, RRRR\} \leq^Q_{\mathbf{V},\mathcal{C}} \{vv, RRRR\}$, and $\{vRR, vv, RRRR\} \geq^Q_{\mathbf{V},\mathcal{C}} \{vv, RRRR\}$, when for example there is a single view $V = \{RR\}$, with v as the corresponding representative view symbol, and $\mathcal{C} = \emptyset$. However, if we define $Q_1 \equiv^Q_{\mathbf{V},\mathcal{C}} Q_2$ iff $Q_1 \leq^Q_{\mathbf{V},\mathcal{C}} Q_2$ and $Q_2 \leq^Q_{\mathbf{V},\mathcal{C}} Q_1$, we get a partial order on the equivalence classes.

Notably, we have that if a set Q' is $\leq^Q_{\mathbf{V},\mathcal{C}}$-maximal, then its equivalence class is a singleton. For this, observe that we cannot replace just any subword which is in some word of some $max_{\mathcal{C}}(V_i)$, for $i \in [1, n]$. However, a word $w = w_1 w_2 w_3$, where $w_2 \in max_{\mathcal{C}}(V_i)$ and $def_{\mathcal{C}}(w_1 v_i w_3) \sqsubseteq_{\mathcal{C}} Q$, is not yet an "optimal" word, and we call w_2 a subword *$\mathcal{C}$-eligible for replacement*. On the other hand, we call a word on $\Omega \cup \Delta$, that has no subwords $\mathcal{C}$-eligible for replacement, a *$\mathcal{C}$-optimal* word.

THEOREM 13. *Let Q' be a $\mathcal{C}$-constrained partial $\mathbf{V}$-rewriting of Q on $\Omega \cup \Delta$. Then, Q' is $\leq^Q_{\mathbf{V},\mathcal{C}}$-maximal, if and only if, there does not exist a non $\mathcal{C}$-optimal word in Q'.*

PROOF. *(If.)* This direction is easy to see because, if there is no word in $max_{\mathcal{C}}(V_i)$, for any $i \in [1, n]$, that appears as a subword $\mathcal{C}$-eligible for replacement in any of the words of Q', then it is impossible to obtain any new $\leq^Q_{\mathbf{V},\mathcal{C}}$-larger rewriting from Q'.

(Only if.) Let's suppose that there are subwords $\mathcal{C}$-eligible for replacement in the words of Q'. Then, for each word $w \in Q'$ compute a word $w_{\mathbf{V}}$ by exhaustively replacing the subwords $\mathcal{C}$-eligible for replacement in w, until nothing can be replaced anymore. If there are no subwords $\mathcal{C}$-eligible for replacement in w, then $w_{\mathbf{V}}$ equals w. Clearly, for each word w there is at least one such word $w_{\mathbf{V}}$, and the number of steps for computing it, is bounded by the length of w. Now, consider the rewriting $Q'' = \bigcup_{w \in Q'} \{w_{\mathbf{V}}\}$. For the rewriting Q'' we have that (a) $Q'' \neq Q'$, (b) $Q' \leq^Q_{\mathbf{V},\mathcal{C}} Q''$, and (c) we cannot obtain any new $\leq^Q_{\mathbf{V},\mathcal{C}}$-larger rewriting from Q''. We

can see that, from (a) and (c) Q' and Q'' cannot belong to the same equivalence class, and from (b), Q'' is $\leq^Q_{\mathbf{V},\mathcal{C}}$-larger than Q'. All the above show that Q' cannot be a maximal rewriting, and this is a contradiction. □

COROLLARY 4. *If Q' is $\leq^Q_{\mathbf{V},\mathcal{C}}$-maximal, then its $\equiv^Q_{\mathbf{V},\mathcal{C}}$-equivalence class is a singleton.*

PROOF. Since Q' is $\leq^Q_{\mathbf{V},\mathcal{C}}$-maximal, from the above theorem we have that, from Q', there cannot be obtained any new $\leq^Q_{\mathbf{V},\mathcal{C}}$-larger rewriting. This means in turn that the equivalence class of Q' is a singleton. □

As discussed in [8, 15, 16], a rewriting Q' is (more) useful for query optimization when it is exact.

A rewriting, that is both $\leq^Q_{\mathbf{V},\mathcal{C}}$-maximal and exact, is the *union* of all $\leq^Q_{\mathbf{V},\mathcal{C}}$-maximal $\mathbf{V}$-rewritings of Q. We call this rewriting the *$\mathcal{C}$-constrained maximal partial $\mathbf{V}$-rewriting* of Q, and denote it with $CMPR_{\mathbf{V}}(Q)$.

From all the above, we can see that $CMPR_{\mathbf{V}}(L)$ is the set of *all* the words on $\Omega \cup \Delta$ with no subword $\mathcal{C}$-eligible for replacement. Formally, we have:

THEOREM 14. *The rewriting $CMPR_{\mathbf{V}}(Q)$ is $\leq^Q_{\mathbf{V},\mathcal{C}}$-maximal and $\mathcal{C}$-constrained exact.*

PROOF. The $\leq^Q_{\mathbf{V},\mathcal{C}}$-maximality follows from the fact that $CMPR_{\mathbf{V}}(Q)$ is the union of the sets of words w with no subwords $\mathcal{C}$-eligible for replacement. For $\mathcal{C}$-constrained exactness, observe that by definition we have $def(CMPR_{\mathbf{V}}(Q)) \sqsubseteq_{\mathcal{C}} Q$. On the other hand, consider a word $w \in Q$. Iterating a finite number of times (function of the length of w) we can find a word $w' \in (\Omega \cup \Delta)^*$ with the smallest possible number of Δ symbols and such that $w \in def_{\mathcal{C}}(w') \sqsubseteq_{\mathcal{C}} Q$. Clearly, $w' \in CMPR_{\mathbf{V}}(Q)$. □

Now, let's consider again the rewritings of [16]. As mentioned before, a $\mathbf{V}$-rewriting of Q is any language Q' on $\Omega \cup \Delta$, such that $def(Q') \subseteq Q$. Notably, the definition of the $\mathbf{V}$-rewritings of Q is compliant with the definition of the $\mathcal{C}$-constrained $\mathbf{V}$-rewritings of Q, and the partial order $\leq^Q_{\mathbf{V}}$ is also compliant with the partial order $\leq^Q_{\mathbf{V},\mathcal{C}}$. Formally speaking, $\leq^Q_{\mathbf{V}}$ is a restriction of $\leq^Q_{\mathbf{V},\mathcal{C}}$. This is because any (algebraically contained) rewriting in [16], by the following theorem, is also a $\mathcal{C}$-constrained rewriting.

THEOREM 15. *Let Q' be a $\mathbf{V}$-rewriting of Q as defined in [16]. Then, Q' is also a $\mathcal{C}$-constrained $\mathbf{V}$-rewriting of Q.*

Consider now the maximal partial rewriting $MPR_{\mathbf{V}}(Q)$ in [16]. As mentioned before, $MPR_{\mathbf{V}}(Q)$ is defined as the union of all $\leq^Q_{\mathbf{V}}$-maximal $\mathbf{V}$-rewritings of Q, and in simple words this means that $MPR_{\mathbf{V}}(Q)$ is the set of all "optimal" words w on $\Omega \cup \Delta$, such that $def(w) \subseteq Q$. Naturally, a word $w = w_1w_2w_3$, where $w_2 \in V_i$ and $def(w_1v_iw_3) \subseteq Q$, for some $i \in [1, n]$, is not yet an "optimal" word, and we call w_2 a subword *eligible for replacement*. On the other hand, we call a word on $\Omega \cup \Delta$, that has no subwords eligible for replacement, an *optimal* word.

Since $\leq^Q_{\mathbf{V}}$ is a restriction of $\leq^Q_{\mathbf{V},\mathcal{C}}$, we have that, in general, $MPR_{\mathbf{V}}(Q) \leq^Q_{\mathbf{V},C} CMPR_{\mathbf{V}}(Q)$, and this means that, in the presence of constraints, $CMPR_{\mathbf{V}}(Q)$ is always a better (never worse) rewriting than $MPR_{\mathbf{V}}(Q)$. Finally, when the set $\mathcal{C}$ of constraints is empty, $CMPR_{\mathbf{V}}(Q)$ coincides with $MPR_{\mathbf{V}}(Q)$.

We will now give a characterization for computing the rewriting $CMPR_{\mathbf{V}}(Q)$. Namely, we show that we can compute it by a language theoretic construction. Let $\mathbf{V} = \{V_1, \ldots, V_n\}$. Then $max_{\mathcal{C}}(\mathbf{V}) = \{max_{\mathcal{C}}(V_1), \ldots, max_{\mathcal{C}}(V_n)\}$. Consider for $max_{\mathcal{C}}(\mathbf{V})$ the same alphabet Ω of view symbols.

THEOREM 16. $CMPR_{\mathbf{V}}(Q) = MPR_{max_{\mathcal{C}}(\mathbf{V})}(max_{\mathcal{C}}(Q))$.

PROOF. *($\subseteq$-direction).* Let $w \in CMPR_{\mathbf{V}}(Q)$. For w we have that $def_{\mathcal{C}}(w) \sqsubseteq_{\mathcal{C}} Q$, and since $max_{\mathcal{C}}(Q)$ is the $\subseteq$-largest $\mathcal{C}$-equivalent language on Δ, we have that $def_{\mathcal{C}}(w) \subseteq max_{\mathcal{C}}(Q)$. This is the first condition for a word to be in a $max_{\mathcal{C}}(\mathbf{V})$-rewriting of $max_{\mathcal{C}}(Q)$. Additionally, we should show that w is optimal with respect to the sets $max_{\mathcal{C}}(\mathbf{V})$ and $max_{\mathcal{C}}(Q)$. Let's suppose that w is not optimal. This means in turn that w can be written as $w = w_1w_2w_3$, where $w_2 \in max_{\mathcal{C}}(V_i)$ and $def_{\mathcal{C}}(w_1v_iw_3) \subseteq max_{\mathcal{C}}(Q)$, for some $i \in [1, n]$. This fact, in other words, means that w is not $\mathcal{C}$-optimal and so, it cannot belong to $CMPR_{\mathbf{V}}(Q)$, which is a contradiction.

($\supseteq$-direction). Let $w \in MPR_{max_{\mathcal{C}}(\mathbf{V})}(max_{\mathcal{C}}(Q))$. For w we have that $def_{\mathcal{C}}(w) \subseteq max_{\mathcal{C}}(Q)$, which implies that $def_{\mathcal{C}}(w) \sqsubseteq_{\mathcal{C}} Q$. This is the first condition for a word to be in a $\mathcal{C}$-constrained $\mathbf{V}$-rewriting of Q. Additionally, we should show that w is $\mathcal{C}$-optimal. Let's suppose that w is not $\mathcal{C}$-optimal. This means in turn that w can be written as $w = w_1w_2w_3$, where $w_2 \in max_{\mathcal{C}}(V_i)$ and $def_{\mathcal{C}}(w_1v_iw_3) \sqsubseteq_{\mathcal{C}} Q$, for some $i \in [1, n]$. Similarly as in the first part of the proof, this implies $def_{\mathcal{C}}(w_1v_iw_3) \subseteq max_{\mathcal{C}}(Q)$. Finally, the last containment says that in w there are still subwords eligible for replacement, and so w cannot belong to $MPR_{max_{\mathcal{C}}(\mathbf{V})}(max_{\mathcal{C}}(Q))$, which is a contradiction. □

Based on the above theorem and Theorem 3, we have the following corollary.

COROLLARY 5. *If $\mathcal{C}$ is a set of word constraints, then $CMPR_{\mathbf{V}}(Q) = MPR_{anc_{\mathcal{C}}(\mathbf{V})}(anc_{\mathcal{C}}(Q))$, where $anc_{\mathcal{C}}(\mathbf{V}) = \{anc_{\mathcal{C}}(V_1), \ldots, anc_{\mathcal{C}}(V_n)\}$.*

A set $\mathcal{C}$ of constraints is *regularity preserving* when, for any regular language L on Δ, $max_{\mathcal{C}}(L)$ is regular as well. Similarly, $\mathcal{C}$ is *context-freeness preserving* when, for any context-free language L on Δ, $max_{\mathcal{C}}(L)$ is context-free as well.

Now, a first conclusion from Theorem 16 is that, when $\mathcal{C}$ is regularity preserving, $CMPR_{\mathbf{V}}(Q)$ can be effectively computed by the algorithm given in [16].

We can raise the question about rewriting using views when the set $\mathcal{C}$ of constraints is not regularity preserving. We show that in such cases, even when $\mathcal{C}$ is context-freeness preserving, it is undecidable in general to test the existence of a "useful" rewriting using views. We define a rewriting as *useful* when it contains at least one word, which has at least one view symbol in it.

THEOREM 17. *There exist a query Q, and a set $\mathbf{V}$ of views, such that the existence of a (constrained) useful rewriting of Q by $\mathbf{V}$ is undecidable for the class of context-freeness preserving constraints.*

PROOF. We give a reduction from the universality problem for the context free class CFG of grammars. Let $G = (\Gamma, \Delta, S, \Pi)$ be a grammar in CFG, with Γ and Δ being its sets of non-terminals and terminals respectively ($\Gamma \cap \Delta = \emptyset$), $S \in \Delta$ being the start symbol, and Π being the set of production rules

$$\{(u_i, t_i) : u_i \in \Gamma, t_i \in (\Delta \cup \Gamma)^*, \text{ for } i \in [1, m]\}.$$

Clearly, for the set $\mathcal{C} = \{t_i \sqsubseteq u_i : i \in [1, m]\}$ of word constraints on $\Delta \cup \Gamma$, we have that $L(G) = anc_{\mathcal{C}}(S) \cap \Delta^*$. Also, for any symbol $T \in \Gamma$, $anc_{\mathcal{C}}(T)$ is context-free since it is the set of all the sentential forms of a context-free sub-grammar. On the other hand, for any symbol $R \in \Delta$, $anc_{\mathcal{C}}(R) = \{R\}$. Since the context free languages are closed under union, concatenation and Kleene star, we conclude that the ancestor function $anc_{\mathcal{C}}$ applied to regular languages on $\Delta \cup \Gamma$ does not escape from the class of context-free languages. Since by Theorem 3, $max_{\mathcal{C}} = anc_{\mathcal{C}}$, we have that $\mathcal{C}$ is context-freeness preserving.

Let \$ be a special symbol not in $\Delta \cup \Gamma$. We take the database alphabet to be $\Delta \cup \Gamma \cup \{\$\}$, $Q = \{\$S\$\}$, $\mathcal{C}$ as above, and a single view $V = \$\Delta^*\$$. From Corollary 5, $CMPR_{\mathbf{V}}(Q) = MPR_{anc_{\mathcal{C}}(\mathbf{V})}(anc_{\mathcal{C}}(Q))$, where $anc_{\mathcal{C}}(\mathbf{V}) = \{anc_{\mathcal{C}}(V)\}$. We observe that $anc_{\mathcal{C}}(V) = anc_{\mathcal{C}}(\{\$\Delta^*\$\}) = V$. So, in fact $MPR_{anc_{\mathcal{C}}(\mathbf{V})}(anc_{\mathcal{C}}(Q)) = MPR_{\mathbf{V}}(anc_{\mathcal{C}}(Q))$. Now, because of the special symbol \$, testing if there is a useful constrained rewriting is equivalent with testing if $MPR_{\mathbf{V}}(anc_{\mathcal{C}}(Q)) \cap \Omega \neq \emptyset$. The last can happen if and only if $V \subseteq anc_{\mathcal{C}}(Q)$. This is $\$\Delta^*\$ \subseteq anc_{\mathcal{C}}(\$S\$) = \$\, anc_{\mathcal{C}}(S)\$$, which is equivalent to $\Delta^* \subseteq anc_{\mathcal{C}}(S)$. Finally, $\Delta^* \subseteq anc_{\mathcal{C}}(S)$ is equivalent to $\Delta^* \subseteq anc_{\mathcal{C}}(S) \cap \Delta^*$, which is nothing else but $\Delta^* \subseteq L(G)$, i.e. $\Delta^* = L(G)$ since $L(G)$ is a pure Δ language. □

Let us now analyze the complexity of computing the rewriting $CMPR_{\mathbf{V}}(Q)$, when $\mathcal{C}$ is a prefix overlapping set of word constraints. As shown in Section 4, for any regular language L on Δ, we can polynomially compute its language of ancestors $anc_{\mathcal{C}}(L)$, which is regular as well. Then, by Corollary 5, we conclude that the computation of $CMPR_{\mathbf{V}}(Q)$ remains in the same complexity class as that of $MPR_{\mathbf{V}}(Q)$. The lower bound can be established by the fact that in the absence of constraints, i.e. when $\mathcal{C} = \emptyset$, $CMPR_{\mathbf{V}}(Q)$ coincides with $MPR_{\mathbf{V}}(Q)$.

6. REFERENCES

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

[2] S. Abiteboul, V. Vianu. Regular Path Queries with Constraints. *Journal of Computing and System Sciences* 58(3) 1999, pp. 428-452

[3] L. Boasson, M. Nivat. Centers of Languages. *Proc. of Theoretical Computer Science, 5th GI-Conference* 1981, LNCS 104, pp. 245-251.

[4] R. Book, F. Otto *String Rewriting Systems* Springer Verlag, 1993.

[5] P. Buneman. W. Fan, S. Weinstein. Path Constraints in Semistructured and Structured Databases. *Proc. of PODS* 1998, pp. 129-138.

[6] P. Buneman. W. Fan, S. Weinstein. Query Optimization for Semistructured Data Using Path Constraints in a Deterministic Data Model. *Proc. of DBPL* 1999, pp. 208-223.

[7] D. Calvanese, G. Giacomo, M. Lenzerini and M. Y. Vardi. Rewriting of Regular Expressions and Regular Path Queries. *Proc. of PODS* 1999, pp. 194-204.

[8] D. Calvanese, G. Giacomo, M. Lenzerini and M. Y. Vardi. Answering Regular Path Queries Using Views. *Proc. of ICDE* 2000, pp. 389-398.

[9] D. Calvanese, G. Giacomo, M. Lenzerini and M. Y. Vardi. View-Based Query Processing for Regular Path Queries with Inverse. *Proc. of PODS* 2000, pp. 58-66.

[10] D. Caucal. On the Transition Graphs of Turing Machines. *Proc. of Machines, Computations, and Universality, Third Int'l Conf.* 2001, pp. 177-189

[11] A. Deutsch, V. Tannen. Optimization Properties for Classes of Conjunctive Regular Path Queries. *Proc. of DBLP* 2001, pp. 21-39.

[12] D. Florescu, A. Y. Levy, D. Suciu Query Containment for Conjunctive Queries with Regular Expressions *Proc. of PODS* 1998, pp. 139-148.

[13] G. Grahne and A. O. Mendelzon. Tableau Techniques for Querying Information Sources through Global Schemas. *Proc. of ICDT* 1999 pp. 332-347.

[14] G. Grahne and A. Thomo. An Optimization Technique for Answering Regular Path Queries. *Proc. of WebDB* 2000.

[15] G. Grahne and A. Thomo. Algebraic Rewritings for Optimizing Regular Path Queries. *Proc. of ICDT* 2001 pp. 301-315.

[16] G. Grahne and A. Thomo. New Rewritings and Optimizations for Regular Path Queries. *Proc. of ICDT* 2003. pp. 242-258.

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

[18] A. Y. Levy. *Answering queries using views: a survey.* Technical Report, Comp. Sci. Dept., Washington Univ., 2000.

[19] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, D. Srivastava. Answering Queries Using Views. *Proc. of PODS* 1995, pp. 95-104.

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

[21] A. O. Mendelzon, G. A. Mihaila and T. Milo. Querying the World Wide Web. *Int. J. on Digital Libraries 1(1)*, 1997 pp. 54-67.

[22] T. Milo and D. Suciu. Index Structures for Path Expressions. *Proc. of ICDT*, 1999, pp. 277-295.

[23] Y. Papakonstantinou, V. Vassalos. Query Rewriting for Semistructured Data. proc. of SIGMOD 1999, pp. 455-466

[24] G. Senizergues. Some Decision Problems about Controlled Rewriting Systems. *Theoretical Computer Science 71(3)*, 1990, pp. 281-346

[25] M. Sipser. *Introduction to the Theory of Computation* PWS Pub. Co., 1996.

[26] S. Yu. Regular Languages. In: *Handbook of Formal Languages.* G. Rozenberg and A. Salomaa (Eds.). Springer Verlag 1997, pp. 41-110