The View Selection Problem for Regular Path Queries

Sergey Afonin*

Lomonosov Moscow State University, Russia
serg@msu.ru

Abstract. The view selection problem consists of finding a set of views to materialize that can answer the given set of workload queries and is optimal in some sense. In this paper we study the view selection problem for regular path queries over semistructured data and two specific view-based query rewriting formalisms, namely single-word and arbitrary regular rewritings. We present an algorithm that for a given finite set of workload queries, i.e. for a set of regular languages, computes a set of views that can answer every query in the workload and has minimal possible cardinality. If, in addition, a database instance is given then one can construct a viewset such that its size, i.e. amount of space required to store results, is minimal on the database instance.

Keywords: view selection problem, regular path queries, semigroups of regular languages.

1 Introduction

The problem of view based query processing plays an important role in many database applications, including information integration, query optimization, mobile computing and data warehousing. In its general form, the problem is stated as follows. Given a query over database schema and a set of materialized views over the same schema (i.e. a set of queries with precomputed answers – *view extensions*), is it possible to answer the query, completely or partially, using answers to the views? This question has been intensively studied for various data models and different assumptions on views semantic (e.g., [13,11,5,20]). The main approaches to view based query processing are *query rewriting* and *query answering* (see [13] for a survey). In query rewriting approach, given a query Q written in language $\mathcal{Q}$, and a set of views $\mathbf{V}$ that are written in language $\mathcal{V}$ one should construct a rewriting R, in language $\mathcal{R}$, such that $Q(D) = R \circ \mathbf{V}(D)$ for each database instance D. It is worth noticing that query rewriting does not depend on view extensions and can be thought as an algorithm that describes how result of the query can be computed form the views. In contrast, query answering consists of direct computing of all answers to Q from the view extensions. The

* This work has been supported in part by the Russian Federal Agency for Science and Innovations under grant 02.514.11.4078 and by the Russian Foundation for Basic Research under grant 08-01-00882-a.

E.S. Laber et al. (Eds.): LATIN 2008, LNCS 4957, pp. 121–132, 2008.

connection between query rewriting and query answering studied in [20, 4]. In this paper we follow query rewriting approach and we shall say that a viewset $\mathbf{V}$ *answers* a query Q if there exists a rewriting of Q.

The ability to answer a query using *only* the answers to views can significantly decrease query evaluation time. For example, in a mobile computing environment the application does not need to access the network in order to process user's query. If the views can not completely answer a query, they can, nevertheless, speed up query processing, because some computations needed for the query processing may have been done while computing the views. From the other hand, view maintaining can be computationally expensive, and the amount of space that is available to store view results could be bounded. These constraints lead to the natural optimization problem, known as the *view selection problem*, which is, roughly speaking, asking for a set of views that can answer given queries and satisfies specified constraints [6, 21, 18]. A dual problem is the *view optimization* problem [17], which asks for a viewset that satisfies specified constraints and has the same expressive power as a given viewset (i.e., answer every query that is answered by a given viewset). Possible applications of these problem include view selection and optimization in data management systems, intelligent data placement, minimization of communication costs in distributed systems, optimization of bulk query processing and others (see e.g. [6, 7, 21, 18, 17]).

The view selection problem was studied in various settings for relational databases, see e.g. [6, 7], and for XPath queries [21, 18]. In this paper we consider the view selection problems for *regular path queries* over semistructured data. Informally, a database is an edge-labelled directed graph, and a query is a regular language over a finite alphabet Σ (without loss of generality we can assume that Σ is the set of all possible labels of a database graph). The result of a query Q is the set of all pairs (u, v) of graph vertices such that there exists at least one labeled path between u and v in the graph and its labels comprise a word in Q. Although the main topic of current research on semistructured data has shifted toward tree data models and corresponding query languages graph model is important in data integration domain. This model is quite flexible and lays between full text search and XML-like tree data models [9]. Regular path query evaluation has polynomial time complexity with respect to both database and query size (one should check non-emptiness of the intersection of some regular languages), but it can be expensive for large databases because the whole database graph may need to be searched, which is inefficient. Materialized views may decrease query processing time drastically.

For regular path query rewritings the so called *complete* [3] and *partial* [10] rewritings were proposed. In both cases a rewriting of a query Q over Σ with respect to a set of views $\mathbf{V}$ is a regular language R over a fresh alphabet Δ, such that a substitution of languages over Σ instead of corresponding letters of Δ yields the original language Q. The two approaches differ in possible mappings for letters of Δ. In complete rewriting Δ letters are mapped into elements of $\mathbf{V}$ only, while in partial rewriting a Δ-letter can be mapped into a Σ-letter as well. For example, $R_1 = \delta_1^*$ is a complete rewriting of $Q_1 = a^*$, and $R_2 = \delta_1 c b^* + \delta_1 c \delta_2$

is a partial rewriting of $Q_2 = a^*cb^*$ with respect to the views $\mathbf{V} = \{a^*, b^*\}$ and the natural substitution $\delta_1 \mapsto a^*$, $\delta_2 \mapsto b^*$. In general, both rewriting techniques may be considered as special cases of the *language substitution problem* which is stated as follows. Given a regular language Q over a finite alphabet Σ and a regular language substitution φ from finite set Δ into regular languages over Σ one should find a language R over Δ, such that $\varphi(R) = Q$. Additional constraints on the structure of language R may be posed. It is known [14] that for any finite set $\mathbf{V}$ of regular languages over Σ, and the subset $T \subseteq \{\cdot, \cup, ^*\}$ of language operations (concatenation, union, and iteration) it is decidable whether or not the regular language Q may be constructed from elements of $\mathbf{V}$ using a finite number of operations from T. Note, that if all three operations are allowed then we have complete rewriting problem mentioned above. The algorithm for partial rewriting construction [10] computes the exhaustive rewriting that, roughly speaking, uses available views as much as possible. In the above example this algorithm gets the rewriting $R'_2 = \delta_1 c \delta_2$ but not $R_2 = \delta_1 cb^* + \delta_1 c \delta_2$.

In this paper we deal with two types of complete rewritings: a *single word rewriting* (the subset T contains only concatenation), and an arbitrary *regular rewriting*. Clearly, that if there exists a single word rewriting of a query, then there exists an arbitrary rewriting of this query with respect to the same set of views. Thus, single word rewritings have weaker expressive power then arbitrary rewritings. Practical meaning of single word rewritings can be justified by some redundancy presented in arbitrary regular rewritings. Known algorithms for regular rewritings of a query Q [3, 14] compute the so-called *maximal* rewriting, which is the set of all words over Δ, such that the substitution of corresponding languages yields a subset of Q. For instance, in the previously considered example the maximal rewriting is $R = \delta_1^*$, while more efficient single word rewriting $R' = \delta_1$ exists. In this case the single word rewriting R' shows that the query can be answered by the views without any computations at all. In order to compute the answer using maximal rewriting R the transitive closure of the view should be constructed. Another attractive property of single word rewritings is that they does not contain recursion (i.e. Kleene star) and admit efficient processing algorithms. Finally, the set of all single word rewritings is an effectively constructable regular language [2].

For now, let us consider the problems under investigation. Assume that a database *workload*, i.e. the set $\mathbf{Q}$ of the most popular queries, is known. What views should be materialized in the database in order to speed up these queries? Trivial solution when every query from the workload correspond to a view is not practical because view maintaining is computationally expensive. In this respect, view selection should be based not only on the ability to answer queries from the workload but on some "efficiency" measure as well. Possible measures are *cost of workload queries evaluation*, *storage constraints*, *cardinality of a viewset*, and *efficiency of rewriting*. We consider the following specific problems (for both single word and regular rewritings).

- *Viewset of minimal cardinality.* What is the minimal number of views that should be materialized in order to answer every query from the workload?

- *Instance based minimal size viewset.* If a particular database instance is given, what is the minimal amount of space required to store the results of a viewset (over this instance) that answer every query from the workload?
- *Oracle based minimal size viewset.* Assume that there is an oracle that for every regular path query estimates the size of the result. What is the minimal size of a viewset that answers queries from the workload?
- *p-Containment of viewsets.* Given two viewsets $\mathbf{V}_1$ and $\mathbf{V}_2$ is it decidable whether or not every query Q that can be rewritten in terms of $\mathbf{V}_1$ can be rewritten in terms of $\mathbf{V}_2$?

It is worth noticing that in some cases (e.g., in data integration systems based on local-as-view approach) there exist a set of views that can not be changed (the views that describe information sources). Nevertheless, database applications are allowed to define "top-level" views that are subject to optimization. Our contribution is the following.

- First, we prove that the search space for minimal cardinality and instance based minimal size viewset construction problems is finite, both for single word and regular rewritings. Thus, both problems are decidable and we propose the algorithms for such viewsets construction.
- We prove that p-containment of viewsets of regular path queries is decidable for single word and regular rewritings.
- We present an algorithm which for a given workload and a database instance computes a viewset of minimal size (on this particular database instance).

The results on single-word rewritings may be considered as a contribution to formal language theory. In particular, we prove that the rank problem for finitely generated semigroups of regular languages is decidable.

The structure of this paper is the following. In Section 2 we introduce basic definition and known results on regular path query rewritings. Algorithms for minimal cardinality viewset construction, for both single word and regular rewritings, are presented in Section 3. This section also contains the the algorithm for checking p-containment of viewsets, for both types of rewritings. Minimal size viewset problem is discussed in Section 4.

2 Regular Path Query Rewritings

In order to fix the notation we start this section with basic notions of formal languages.

An *alphabet* is a finite non-empty set of *symbols*. A finite sequence of symbols from an alphabet Σ is called a *word* in Σ. The *empty* word is denoted by ε. Any set of words is called a *language* over Σ. Σ^* denotes the set of all words (including the empty word) in a given alphabet, Σ^+ denotes the set of all non-empty words in Σ, $\varnothing$ is the empty language (containing no words), and 2^{Σ^*} is the set of all languages over Σ.

Let u be a word in Σ and $A \subseteq \Sigma^*$. The right quotient of A with respect to u is the language $u^{-1}A = \{v \in \Sigma^* \mid uv \in A\}$, and the left quotient is $Au^{-1} = \{v \in \Sigma^* \mid vu \in A\}$. If $A = \{w\}$ we shall write $u^{-1}w$ instead of $u^{-1}\{w\}$.

The *union* of languages L_1 and L_2 is the language $L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \vee w \in L_2\}$. The language $L_1L_2 = \{w \in \Sigma^* \mid \exists w_1 \in L_1, w_2 \in L_2 : w = w_1w_2\}$ is called a *concatenation* of L_1 and L_2. $L^k = LL^{k-1}$ is L to the power k. By definition, L to the power zero is the empty word: $L^0 = \{\varepsilon\}$. The Kleene closure (or star) of L is the language $L^* = \cup_{k=0}^{\infty} L^k$. A language is *regular* if it can be obtained from singleton languages (i.e., the letters of the alphabet), the empty language, and $\{\varepsilon\}$, using a finite number of operations of concatenation, union and closure. The set of all regular languages over an alphabet Σ is denoted by $\mathrm{Reg}(\Sigma)$. In the sequel all the languages are assumed regular.

Let Δ be an alphabet and S be a semigroup. A morphism $\varphi : \Delta^+ \to S$ is any function satisfying $\varphi(uv) = \varphi(u)\varphi(v)$ for all $u, v \in \Delta^+$. A morphism $\varphi : \Delta^+ \to 2^{\Sigma^*}$ is called a *regular language substitution* if $\varphi(\delta)$ is a regular language over Σ for all $\delta \in \Delta$. Every regular language substitution $\varphi : \Delta^+ \to \mathrm{Reg}(\Sigma)$ generates the semigroup $\mathcal{S}_\varphi = \langle\{\varphi(\delta) \mid \delta \in \Delta)\}\rangle$. Conversely, with every semigroup $(\mathcal{S}, \cdot) = \langle V_1, \ldots, V_n\rangle$ of regular languages we can associate a language substitution φ defined by the rule $\delta_i \mapsto V_i$.

We turn now to regular path queries over semistructured data and query rewritings. A *semistructured database* is an ordered edge-labeled graph $\mathcal{B} = \langle O, E, \Sigma, \psi\rangle$, where O is a finite set of nodes (objects), Σ is a set of labels (a database alphabet), $E \subseteq O \times O$ is a set of edges, and $\psi : E \to \Sigma$ is the edge-labeling function. With every path in the database graph, i.e. a set of adjusting edges $(e_1, e_2, \ldots, e_m)$, we associate the word $w = a_1a_2 \ldots a_m$ in Σ, defined by the rule $a_i = \psi(e_i)$. Let us denote the regular language of all words associated with all paths between given pair of database nodes, say u and v, as $L_{(u,v)}(\mathcal{B})$. A *query* is a regular language over Σ. The *result* of a query Q on the database $\mathcal{B}$ is the set

$$Q(\mathcal{B}) = \{(u, v) \in O \times O \mid L_{(u,v)}(\mathcal{B}) \cap Q \neq \varnothing\}.$$

Let $\varphi : \Delta^+ \to \mathrm{Reg}(\Sigma)$ be a regular language substitution. The *maximal rewriting* of a regular language $R \subseteq \Sigma^*$ with respect to φ is the set

$$M_\varphi(R) = \{w \in \Delta^+ \mid \varphi(w) \subseteq R\}.$$

The maximal rewriting $M_\varphi(R)$ is called *exact*, if

$$\bigcup_{w \in M_\varphi(R)} \varphi(w) = R.$$

The following theorem is due to D.Calvanese et al. [3], and K.Hashiguchi [14].

Theorem 2.1. *Let $\varphi : \Delta^+ \to \mathrm{Reg}(\Sigma)$ be a regular language substitution. For any regular language $R \subseteq \Sigma^*$ the maximal rewriting $M_\varphi(R)$ is a regular language over Δ.*

As it was mentioned above, the maximal rewriting could be redundant. We say, that a regular language Q admits a *single word rewriting* with respect regular language substitution φ if there exists a word $w \in \Delta^+$, such that $Q = \varphi(w)$. The decidability of single word rewriting problem was proved by K.Hashiguchi [14] and the following theorem was proved in [2].

Theorem 2.2. *Let $\varphi : \Delta^+ \to \mathrm{Reg}(\Sigma)$ be a regular language substitution and w be a word in Δ^+. The membership problem for the semigroup $\mathcal{S}_\varphi$*

$$[w] = \{u \in \Delta^+ \mid \varphi(u) = \varphi(w)\}$$

is a regular language over Δ.

Theorem 2.2 implies that, given a finite set of views and a regular path query Q, one can effectively construct a finite automaton that recognize the set of all possible single word rewritings of Q wrt the viewset.

3 Viewsets of Minimal Cardinality and p-Containment

In this section we study the problem of finding a viewset of minimal cardinality and show decidability of p-containment between viewsets of regular path queries. We focus our attention on single word rewritings and then extend the result to arbitrary regular rewritings. For single word rewritings both problems may be reformulated in terms of finitely generated semigroups of regular languages. Indeed, given a finite set $\mathbf{V}$ of regular languages over Σ one can consider the semigroup $\mathcal{S} = \langle \mathbf{V}, \cdot \rangle$ with $\mathbf{V}$ as the set of generators and language concatenation as a semigroup product. The semigroup $\mathcal{S}$ consists exactly of languages that can be rewritten in terms of $\mathbf{V}$ using single word rewriting. Now, a viewset $\mathbf{V}_1$ is p-contained in a viewset $\mathbf{V}_2$ iff the semigroup $\langle \mathbf{V}_1, \cdot \rangle$ is a subsemigroup of $\langle \mathbf{V}_2, \cdot \rangle$, and minimal cardinality viewset problem is the following. Given a finite set $\mathbf{Q} = \{Q_1, \ldots, Q_n\}$ of regular languages over an alphabet Σ one should find a natural number k and a set $\mathbf{G} = \{G_1, \ldots, G_k\}$ of regular languages over Σ satisfying the following conditions:

- $Q_i \in \langle \mathbf{G} \rangle$ for every $i \in \{1, \ldots, n\}$, and
- for every $k' < k$ and every set $\mathbf{G}' = \{G'_1, \ldots, G'_{k'}\}$ there exists $Q \in \mathbf{Q}$ such that $Q \notin \langle \mathbf{G}' \rangle$.

The above conditions imply that $\langle \mathbf{Q} \rangle \subseteq \langle \mathbf{G} \rangle$. Note, that k is bounded by the number of elements in $\mathbf{Q}$ and set $\mathbf{G}$ always exists, although it can not be constructed by a "trivial" brute force algorithm.

The main idea of the algorithm proposed in this section is to construct factorizations of Q_i (with respect to concatenation) and choose minimal subset of factors that forms a basis of a semigroup. The difficulty related to this approach arise from the fact that a regular language has infinitely many factorizations, in general. For example, the language $L = a^*$ admits infinitely many factorizations of the form $L = FM$, e.g., every pair of languages F and M such that $F \cup M = a^*$ and $\varepsilon \in F \cap M$ forms a factorizations of L. In order to resolve this problem we show that there exist finite set of languages, that can be effectively constructed from $\mathbf{Q}$, and minimal basis consist of elements of this set.

3.1 Maximal Factors of Regular Languages

Let $L \subseteq \Sigma^*$ be a regular language. Consider a family of *language equations* of the form $L = X_1 \dots X_m$. A m-tuple $(L_1, \dots, L_m)$ of languages is a solution to the equation $L = X_1 \dots X_m$ if the equality $L = L_1 \dots L_m$ holds. A m-tuple $(L_1, \dots, L_m)$ is contained in a m-tuple $(L'_1, \dots, L'_m)$ if $L_i \subseteq L'_i$ for all $i = 1, \dots, m$. A solution $(L_1, \dots, L_m)$ is called *maximal* if it is not contained in any other solution to the equation. A language F is called a *maximal factor* of L if there exist a natural number m such that F is a component of some maximal solution to the equation $L = X_1 \dots X_m$. The set of all maximal factors of a language L is denoted by $\mathcal{F}(L)$.

Theorem 3.1 (J.Conway [8]). *A regular language L has finitely many maximal factors and all these factors are regular languages. Moreover, maximal factors are effectively constructable.*

The algorithm for viewset selection depends on maximal factors construction and we briefly describe how this can be done. First, consider the univariate language equation of the form $AY = L$ where $A, L \in \mathrm{Reg}(\Sigma)$. A language $F \subseteq \Sigma^*$ is called a *solution* to the equation $AY = L$ if $AF = L$. Since the union of two solutions is a solution as well, the equation $AY = B$ has at most one maximal solution and this solution is the language $Y_{\max} = \bigcap_{u \in A} u^{-1}L$. Similarly, the language $X_{\max} = \bigcap_{u \in A} Lu^{-1}$ is the maximal solution to the equation $XA = B$. Since every regular language has finitely many different right (left) quotients the equation $XY = L$ has at most finitely many maximal solutions.

Let (P_i, Q_i) $i = 1, \dots, p$ be the maximal solutions to the equation $XY = L$. The set of maximal factors of L is exactly the union of $\{P_i\}$, $\{Q_i\}$, and the set of maximal solutions of equations $P_i X Q_j = L$. Maximal solutions to such equations are to intersection of some left and right quotients of L.

3.2 Algorithm

Let $\mathbf{A}$ be a set of languages over Σ. By *intersection closure* of $\mathbf{A}$, denoted as $\overline{\mathbf{A}}$, we mean the set of all languages that can be obtained by finite intersections of elements of $\mathbf{A}$:

$$\overline{\mathbf{A}} = \{I \subseteq \Sigma^* \mid \exists A_1, \dots, A_m \in \mathbf{A}\ I = A_1 \cap \dots \cap A_m\}.$$

Theorem 3.2. *Let $\mathbf{Q}$ be a finite set of regular languages and $\mathbf{F}$ be the set of all maximal factor of its elements, i.e.*

$$\mathbf{F} = \cup_{Q \in \mathbf{Q}} \mathcal{F}(Q).$$

There exists a minimal cardinality set $\mathbf{G}$ of generators for $\mathbf{Q}$ such that $\mathbf{G} \subseteq \overline{\mathbf{F}}$.

Proof. Let A and B be regular languages. We call a language F a *common factor* of A and B if $A = P_1 F P_2$, $B = Q_1 F Q_2$ for some languages P_1, P_2, Q_1, Q_2. We prove now that if there exits a common factor F for A and B when F can be

extended to the intersection $F' = F_A \cap F_B$ of some maximal factors $F_A \in \mathcal{F}(A)$ and $F_B \in \mathcal{F}(B)$, that is $F \subseteq F'$ and F' is a common factor of A and B.

Indeed, languages (P_1, F, P_2) form a solution to the equation $A = X_1X_2X_3$. By Theorem 3.1 there exists a maximal solution, say (M_1, M_2, M_3), that contains the solution (P_1, F, P_2). It is follows that every tuple (P_1, L, P_2) is a solution to the equation if L satisfy the inequalities $F \subseteq L \subseteq M_2$. Similarly, the solution (Q_1, F, Q_2) is contained in some maximal solution to the equation $B = X_1X_2X_3$, and the intersection of corresponding components of these maximal solutions is the desired language F'. Clearly, that if F is a common factor of languages $A_1, \ldots, A_p$, then there exists a common factor F' that is the intersection of some maximal factors of the languages $A_1, \ldots, A_p$.

Now assume that $\mathbf{G} = \{G_1, \ldots, G_k\}$ is a minimal cardinality set of generators and this set is not contained in $\overline{\mathbf{F}}$. Without loss of generality assume that $F_1 \notin \overline{\mathbf{F}}$. Every language $Q \in \mathbf{Q}$ has a factorization of the form

$$Q = G_{i_1} \ldots G_{i_m},$$

and G_1 is a common factor of some languages from $\mathbf{Q}$. Thus, there exists $G_1' \in \overline{\mathbf{F}}$ such that the set $\mathbf{G}^1 = \{G_1', G_2 \ldots, G_k\}$ is the set of generators for $\mathbf{Q}$. Repeatedly applying this procedure to all elements of $\mathbf{G}$ one can construct the set of generators that has the same cardinality as $\mathbf{G}$. □

In the sequel we shall call a viewset that is a subset of $\overline{\mathbf{F}}$ *the maximal viewset*. It is worth noting that in the proof we did not assume that the "initial" minimal cardinality set consists of regular languages, thus if we drop the restriction that minimal set should contains only regular languages we will not decrease the number of generators.

As an immediate corollary from Theorem 3.2 and the finiteness of the set of maximal factors of a regular language we have

Theorem 3.3. *Minimal cardinality viewset problem is decidable for single word rewriting.*

Proof. Given a finite set $\mathbf{Q} = \{Q_1, \ldots, Q_n\}$ the algorithm first computes the intersection closure $\overline{\mathbf{F}}$ of the set $\mathcal{F}(\mathbf{Q})$ of all maximal factors, and then, for every subset $\mathbf{G}$ of $\overline{\mathbf{F}}$, checks whether or not $\mathbf{G}$ is the set of generators for $\mathbf{Q}$. □

The algorithm proposed by this theorem relies on maximal factor construction and the algorithm that checks that a language is representable as concatenation of given regular languages. The number of maximal factors of a regular language L is at most 2^{2N},where N is the number of states of the minimal deterministic automaton for L. The best known bound for the number of intersection of maximal factors is 2^{N^2} [19]. Thus, the size of set $\overline{\mathbf{F}}$ is double-exponential in the total size of minimal automata for languages in $\mathbf{Q}$. Known algorithms for single word rewritings [1,14] are based on the *limitedness property of distance*

automata which is PSPACE-complete [16,15] and for which only exponential time complexity algorithms are known. The number of states of a distance automaton to be checked is no more then 2^M, where M is the number of states of deterministic automaton for Q. Summing up, this algorithm requires PSPACE-complete problem to be called a triple-exponential number of times.

In the rest of this section we show how the minimal cardinality viewset selection problem can be reduced to the *minimal test length* problem. The crucial point in this reduction is Theorem 2.2. Let φ be a substitution defined by the rule $\varphi(\Delta) = \overline{\mathbf{F}}$. We shall abuse the notation and denote by $[Q]$ the set of all words w over Δ such that $\varphi(w) = Q$. The minimal cardinality viewset can be represented as a subset $B \subseteq \Delta$. For every query Q from the workload $\mathbf{Q}$ one can construct an automaton for $[Q]$, and the set B should satisfy the following condition: $[Q] \cap B^* \neq \varnothing$ for every $Q \in \mathbf{Q}$. For every query Q_i one can effectively construct the finite set of subsets of Δ, say $R_1^{(i)}, \ldots, R_{k_i}^{(i)} \subseteq \Delta$, such that Q_i can be represented in terms of $\varphi(B)$ iff $R_j^{(i)} \subseteq B$ for some $j \leqslant k_i$. These sets correspond to (labels on) simple paths in minimal automaton for $[Q_i]$. The minimal cardinality viewset is the minimal cardinality subset $B \subseteq \Delta$ such that for every Q_i there exists $j \leqslant k_i$ satisfying $R_j^{(i)} \subseteq B$. Although the minimal test length problem is NP-complete such reduction allows to avoid checking every possible subset of the set $\overline{\mathbf{F}}$. Moreover, when membership languages $[Q]$ are constructed for all $Q \in \mathbf{Q}$ the remaining computations use transparent structures and the complexity does not depend on number of states of automata for $\overline{\mathbf{F}}$.

3.3 Arbitrary Rewritings and p-Containment of Viewsets

Similarly to maximal solutions to linear language equations one can consider maximal solutions to equations of the form $r(X_1, \ldots, X_n) = L$ where r is an arbitrary regular expression over alphabet $\Sigma \cup \{X_1, \ldots, X_n\}$. It is known that for every regular language L there exist only finitely many different languages that are components of maximal solutions (see e.g. [19]).

Theorem 3.4. *Let* $\mathbf{Q}$ *be a finite set of regular languages. There exists a finite set of languages* $\mathbf{F}$*, that can be effectively constructed from* $\mathbf{Q}$*, such that at least one minimal cardinality viewset (wrt regular rewriting) is contained in* $\mathbf{F}$*.*

The proof is essentially the same as for Theorem 3.2.

The algorithm described in the previous section may produce several viewsets of minimal cardinality. Different viewsets can be compared by their expressive power. We say that a viewset $\mathbf{V}_1$ is *p-contained* in a viewset $\mathbf{V}_2$ if every query that can be answered using $\mathbf{V}_1$ can be answered using $\mathbf{V}_2$. The following theorem gives an algorithm for p-containment checking.

Theorem 3.5. *A viewset* $\mathbf{V}_1$ *is p-contained in a viewset* $\mathbf{V}_2$ *with respect to regular rewritings iff every view* $V \in \mathbf{V}_1$ *may be rewritten in terms of the viewset* $\mathbf{V}_2$*.*

4 Viewsets of Minimal Size

For now assume that the database instance $\mathcal{B}$ is fixed. Every regular path query defines a (finite) binary relation on the set of the database nodes and we define the *size* of a query Q, denoted by $|Q|$, as the cardinality of the corresponding relation. This quantity is proportional to the amount of space, in bytes, required to store the results to the query. Minimal size viewset selection problem is stated as follows: given a workload $\mathbf{Q}$ and a database instance $\mathcal{B}$ find a viewset $\mathbf{V}$ such that its total size $\sum_{V \in \mathbf{V}} |V|$ is minimal on the database $\mathcal{B}$ (the minimum is taken among all the viewsets that answers every query in the workload). The following proposition states that every minimal size viewset is contained in some maximal viewset.

Proposition 4.1. *Let $\mathbf{V}$ be a minimal size viewset for $\mathbf{Q}$ and $\mathcal{B}$. There exists a viewset $\mathbf{G}$ such that:*

- *for every $V \in \mathbf{V}$ there exists $G \in \mathbf{G}$ satisfying $V \subseteq G$;*
- *every language $G \in \mathbf{G}$ is the intersection of some maximal factors: $\mathbf{G} \subseteq \mathcal{F}(\mathbf{Q})$.*

The proof is straightforward. The main idea of the algorithm for instance based minimal size viewset construction is to start from a maximal viewsets and then minimize their elements by subtracting some regular languages.

Proposition 4.2. *Let $\mathbf{G} = \{G_1, \ldots, G_k\}$ be a maximal viewset for the workload $\mathbf{Q}$, and $\mathcal{B}$ be a database instance. There exists an effectively constructable viewset*

$$\mathbf{G}' = \{G'_1, \ldots, G'_k\}$$

such that the following conditions hold:

- *$G'_i \subseteq G_i$ for all $i = 1, \ldots, k$;*
- *for any viewset $\mathbf{G}'' = \{G''_1, \ldots, G''_k\}$ satisfying the above condition the inequality $|\mathbf{G}'| \leqslant |\mathbf{G}''|$ holds.*

Proof. Let $\mathbf{G} = \{G_1, \ldots, G_k\}$ be a maximal viewset that is not a minimal size viewset. There exist a component $G \in \mathbf{G}$ such that $|\mathbf{G}'| < |\mathbf{G}|$, where $\mathbf{G}'$ is obtained from $\mathbf{G}$ by replacing G with some language $G' \subseteq G$. Clearly, that $G'(\mathcal{B}) \subset G(\mathcal{B})$ and the language

$$G'' = G \setminus \bigcup_{(u,v) \in G(\mathcal{B})} L_{(u,v)}(\mathcal{B})$$

has exact the same size as G. Since the database instance is finite, the number of views to be checked is finite and a minimal size viewset (contained in a given viewset) may be constructed by checking all possible combinations. □

As an immediate corollary of Propositions 4.1 and 4.2 we have:

Theorem 4.1. *There exists an algorithm that, for a given workload* $\mathbf{Q}$ *and a database instance* $\mathcal{B}$*, computes a viewset of minimal size.*

Now, suppose that the database instance is not known, but there exists an oracle $E : \mathrm{Reg}(\Sigma) \to \mathbb{N}$ that estimates the size of a query result. Estimation function E is called *exact* if $E(Q)$ equals to actual query result size. In this case we have implication $Q_1 \subseteq Q_2 \Rightarrow E(Q_1) \leqslant E(Q_2)$. We show that the problem of minimal size (with respect to the oracle E) requires construction of minimal solutions to language equations. Indeed, let $\mathbf{Q} = \{Q_1, Q_2\}$ and the equation $Q_1 X = Q_2$ has a solution. A possible minimal size viewset in this case is a pair $\{Q_1, S\}$, where S is a minimal solution to the equation such that $E(S) \leqslant E(S')$ for every other (minimal) solution S'. The problem is that even such simple linear equation admits uncountably many minimal solution. For example, the equation $\{\varepsilon, a\}X = \{a\}^*$ over the unary alphabet $\Sigma = \{a\}$ has infinitely many minimal solutions: every language $K \subseteq \Sigma^*$ that contains the empty word and satisfies the condition that if K contains the word a^n $(n \geqslant 0)$ then it contains only one of the words a^{n+1} or a^{n+2} is a minimal solution. It is not clear whether or not the minimal size viewset can be computed with respect to exact query size estimation function.

5 Conclusion

In this paper we proved that the search space for minimal cardinality and instance based minimal size viewset construction problems is finite, both for single word and regular rewritings. Thus, both problems are decidable and we proposed algorithms for such viewsets construction. We also proved that p-containment of viewsets of regular path queries is decidable for single word and regular rewritings. The main result on single-word rewritings may be stated in terms of semigroups of regular languages: The rank problem for finitely generated semigroups of regular languages is decidable.

It could be interesting to consider similar problems for partial rewritings. Recursionless (i.e., finite) partial rewritings was recently investigated in [12]. It seems that the main problem for view selection under partial rewriting is to choose an appropriate measure for the viewset. Since partial rewriting admits letters of original alphabet in the rewriting language, the empty viewset is the minimal cardinality and minimal size viewset. Thus, a relevant viewset quality measure should take into account (in addition cardinality and storage constraints) how often a query evaluation algorithm will access the database.

References

1. Afonin, S., Khazova, E.: Membership and finiteness problems for rational sets of regular languages. International Journal of Foundations of Computer Science 17(3), 493–506 (2006)
2. Afonin, S., Khazova, E.: A note on finitely generated semigroups of regular languages. In: Andre, J.M., et al. (eds.) Proceedings of the International Conference "Semigroups and Formal Languages", pp. 1–8. World Scientific, Singapore (2007)

3. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.: Rewriting of regular expressions and regular path queries. Journal of Computer and System Sciences 64, 443–465 (2002)
4. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: View-based query processing: On the relationship between rewriting, answering and losslessness. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 321–336. Springer, Heidelberg (2004)
5. Calvanese, D., Giacomo, G.D., Lenzerini, M., Vardi, M.Y.: Answering regular path queries using views. In: ICDE, pp. 389–398 (2000)
6. Chirkova, R., Halevy, A.Y., Suciu, D.: A formal perspective on the view selection problem. The VLDB Journal 11(3), 216–237 (2002)
7. Chirkova, R., Li, C.: Materializing views with minimal size to answer queries. In: PODS 2003, pp. 38–48. ACM Press, New York (2003)
8. Conway, J.: Regular Algebra and Finite Machines. Chapman and Hall, Boca Raton (1971)
9. Franklin, M., Halevy, A., Maier, D.: From databases to dataspaces: a new abstraction for information management. SIGMOD Rec. 34(4), 27–33 (2005)
10. Grahne, G., Thomo, A.: Algebraic rewritings for optimizing regular path queries. Theoretical Computer Science 296(3), 453–471 (2003)
11. Grahne, G., Thomo, A.: Query containment and rewriting using views for regular path queries under constraints. In: PODS 2003, pp. 111–122. ACM Press, New York (2003)
12. Grahne, G., Thomo, A.: Boundedness of regular path queries in data integration systems. In: Proc. of IDEAS, pp. 85–92. IEEE Computer Society, Los Alamitos (2007)
13. Halevy, A.Y.: Theory of answering queries using views. SIGMOD Record (ACM Special Interest Group on Management of Data) 29(4), 40–47 (2000)
14. Hashiguchi, K.: Representation theorems on regular languages. Journal of computer and system sciences 27, 101–115 (1983)
15. Kirsten, D.: Desert automata and the finite substitution problem. In: Diekert, V., Habib, M. (eds.) STACS 2004. LNCS, vol. 2996, pp. 305–316. Springer, Heidelberg (2004)
16. Leung, H., Podolskiy, V.: The limitedness problem on distance automata: Hashiguchi's method revisited. Theoretical Computer Science 310(1–3), 147–158 (2004)
17. Li, C., Bawa, M., Ullman, J.D.: Minimizing view sets without losing query-answering power. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, pp. 99–113. Springer, Heidelberg (2000)
18. Mandhani, B., Suciu, D.: Query caching and view selection for XML databases. In: Böhm, K., Jensen, C.S., Haas, L.M., Kersten, M.L., Larson, P.-Å., Ooi, B.C. (eds.) VLDB, pp. 469–480. ACM Press, New York (2005)
19. Polák, L.: Syntactic semiring and language equations. In: Champarnaud, J.-M., Maurel, D. (eds.) CIAA 2002. LNCS, vol. 2608, pp. 182–193. Springer, Heidelberg (2003)
20. Segoufin, L., Vianu, V.: Views and queries: determinacy and rewriting. In: PODS 2005, pp. 49–60. ACM Press, New York (2005)
21. Tajima, K., Fukui, Y.: Answering XPath queries over networks by sending minimal views. In: Nascimento, M.A., Özsu, M.T., Kossmann, D., Miller, R.J., Blakeley, J.A., Schiefer, K.B. (eds.) VLDB, pp. 48–59. Morgan Kaufmann, San Francisco (2004)