Approximate Reasoning in Semi-structured Databases

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

Abstract

We give a general framework for approximate reasoning in semi-structured databases. For approximate querying, the user will specify a regular path query, and a regular transducer (or weighted regular expression) for the allowed sequences of elementary "distortions" that keeps a word within an approximation of an original word. The transducer also defines a function for the distance between two words. We show that approximate answers to regular path queries are computable, and give an efficient algorithm for the task.

In our framework for approximate reasoning we also allow data instances to *approximately satisfy* a schema such as a Data Guide. We reduce the problem of approximate satisfaction to the *limitedness problem* in Hashiguchi distance automata. We further show that in the common case where the regular transducer is prefix-closed, the limitedness problem can be solved in polynomial space.

1 Introduction

Almost all the query languages for semi-structured data provide the possibility for the user to query the database through regular expressions. These queries are in essence graph patterns and the answers to the query are subgraphs of the database that match the given pattern [MW95, C+99, GT00, GT01]. For example, for answering the query

$$Q = (\_^* \cdot \mathit{article}) \cdot (\_^* \cdot \mathit{ref} \cdot \_^* \cdot (\mathit{ullman} + \mathit{widom}))$$

one should find all the paths having at some point an edge labelled $\mathit{article}$, followed by any number of other edges then by an edge ref and finally by an edge labelled with ullman or widom.

However, we are often willing to live with structural information that is approximate. In other words the semistructured data represented by a graph database can be an approximation of the real world rather than an exact representation. On the other hand the user herself can have an approximate idea and/or knowledge about the world, and this has as a consequence a need for non exact information to be extracted from the database. In both cases the conclusion is that we need to deal with approximate queries and databases, and give approximate answers to the user queries.

If we consider the database graph to be the Web-graph then the current search engines already deal with approximate matching of specific words or sentences against the HTML text of the nodes. The result is usually ranked with regard to the degree of proximity and then presented to the user. However, consider a scenario as in [MMM97] where the links in the HTML pages are labeled by some predicates and we like to find not only specific HTML pages containing some given text, but also we want these pages to be linked by a path on which the link label sequence conforms to a given language. Current search engines do not give the user the option to approximately query the Web-graph through regular expressions. The same is true also for the query languages for semistructured data; they do provide means to specify paths of edge labels through regular expressions, but they do not have capabilities to specify approximate paths.

The similar problem of finding approximate patterns in sequence databases is treated in depth in [JMM95]. There, Jagadish, Mendelzon and Milo formalized a very powerful rule-based system through which a user can specify the possible allowed transformations of a string to some other string. The rule-based system contains in addition a "control sequence" regular expression that specifies all the allowed sequences of applications of the rules.

However, the power of these transformation rules comes with a price to pay: It is undecidable to say, in the general case, if a string s_1 can be transformed into another string s_2 given a set of transformation rules and a regular expression specifying the possible control sequences. Special, decidable cases, such as star-free rules and "atomic" transformation rules are presented in [JMM95].

It is worth noting here that for the free application of the three classical edit operations, *insert*, *delete*, and *substitute*, which can easily be modeled as rules in the formalism of Jagadish et al. the transformation problem is decidable. Actually, as is the case of edit operations, some rules can be thought to correspond to a transducer. Extending this idea, instead of transformation rules and control sequence expressions we consider as a model for string transformation or *distortion* a *regular transducer*. If, instead of a single string we have a regular language L of strings, then the set of all the possible distortions from the language L will be the set of all transductions of the L words through $\mathcal{T}$.

The motivation for considering transducers as a transformation model is similar to the motivation for using regular expressions for querying recursive graph patterns. Since the queries are recursive in their generality, a recursive mechanism is needed for transforming them. Of course, a rule based formalism equipped with recursive control sequences is also a recursive way to transform queries, but in practice, to achieve decidable optimization problems etc, we are content with a more limited form of recursion. This is in analogy with the fact that we are using regular expressions for matching recursive graph patterns and not the more powerful formalisms such as context-free rule-based grammars. As an example of the use of transducers for transformation models consider the query Q above. Suppose we are willing to substitute *article* by *book* at no cost, and we are willing to substitute *ullman* by *abiteboul* at cost 1, and by *bancilhon* at cost 5. Then the desired distortion transducer can be specified by the following extended regular expression:

$$(\Delta, 0, \Delta)^* \cdot ((ullman, 0, ullman) + (ullman, 1, abiteboul) + (ullman, 5, bancilhon))$$

This regular expression is defined over triplets (R, i, S), where i is the cost of substituting R by S, and $(\Delta, 0, \Delta)$ is a shorthand for $\sum_{R\in\Delta}(R, 0, R)$, with Δ being the underlying (finite) alphabet. It is easy to see that such extended regular expressions exactly correspond to regular transducers.

Now, imagine a user query Q expressed by a regular expression, a database graph DB and (an extended regular expression for) a transducer $\mathcal{T}$, describing the tolerable distortions to the query. Intuitively, first we distort the query Q into Q' and then issue Q' against the database DB. The result will be the approximate answer for the query Q. Hence, the transformation problem for transducers is solvable. [1]

The second part of the paper deals the with the other point of view about the approximate representation of the world through a database. We suppose that we have a perfect description of what the world can be; this is the data guide [B+97, GW97, ABS99]. On the other hand we have non-perfect database instances.

We again suppose that the user specifies a distortion transducer $\mathcal{T}$, through which we can distort the data guide through allowed elementary distortions and then test if the database conforms to the distorted data-guide. If the database indeed conforms to the distorted data guide, we are interested in "how far" the database is from the original data guide. We have then, for each path in the database, a set of words in the data guide that have been transduced by $\mathcal{T}$ into this path. We are interested in finding the closest one. Then, as a quantitative measure of approximate satisfaction, we consider the largest among these closest distances. If this largest distance is bounded by some $k \in \mathbb{N}$ we say that the database k-satisfies the data guide. We prove that the problem of the k-satisfaction cannot be easier than PSPACE. We then present a construction by which the problem of finding the above largest distance is reduced to the problem of limitedness in Hashiguchi distance automata [Has82].

The problem of limitedness in distance automata was shown to be decidable by Hashiguchi in [Has82]. This means in turn that our problem of k-satisfaction is decidable. However, the decision procedure of [Has82] requires exponential space, and the problem of a tighter lower bound is still open [Has00]. We prove that if the distance automata is prefix closed, i. e. if all states are final, then the limitedness problem can be decided in polynomial space. This subclass of distance automata corresponds to a large practical sub case of the k-satisfaction problem, that is when the data guide and the distortion transducer with have all the states final. Indeed, if we examine the origin of a data guide [N+97], we can conclude that the assumption made about the all states being final in the data guide automaton is quite reasonable. In fact, the data guide can be considered as a DFA constructed in a bottom up fashion from a set a sample databases, through the well-known subset construction. But the databases considered as automata always have all their states final, and as a consequence, the corresponding DFA will have all states final. Regarding the assumption that the dis-

[1] Of course, regular languages are closed under transductions, i. e. for any regular language Q and regular transducer $\mathcal{T}$, the language $\mathcal{T}(Q)$ is regular. Why would the user not then write down the expression for $\mathcal{T}(Q)$ directly then, instead of giving Q and T. The first point is that it is not always easy to do. The second point is that the user might previously have issued Q, and wants to "relax" or broaden Q without having to rewrite the whole query. The third point is that the user is interested in receiving the answers *ranked* according to their proximity of the query. If the transduction is folded into the query, the "distance" between words and their transductions is lost.

tortion transducer consists of final states only, we note that any edit transducer fulfills this requirement, as does also for instance any transducer specified by a finite set of weighted substitutions of the form *ullman* $\overset{5}{\mapsto}$ *bancilhon*.

2 Graph Databases, Queries and Approximate Answers

Let Δ be a finite alphabet, called the *database alphabet*. Elements of Δ will be denoted $R, S, T, R', S', \ldots, R_1, S_1, \ldots$, etc. 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. Objects will be denoted $a, b, c, a', b', \ldots, a_1, b_2, \ldots$, and so on. 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 labelled with symbols from Δ. If there is a path labelled $R_1, R_2, \ldots, R_k$ from a node a to a node b we write $a \stackrel{R_1.R_2\ldots R_k}{\longrightarrow} b$.

A *query* Q is a finite or infinite regular language over Δ. Let Q be a query and $DB = (N, E)$ a database. Then the *answer to* Q on DB is defined as

$$\mathrm{ans}(Q, DB) \quad = \quad \{(a, b) : \{a, b\} \subseteq N \text{ and } a \stackrel{W}{\longrightarrow} b \text{ for some } W \in Q\}.$$

A *regular transducer* $\mathcal{T} = (S, I, O, \tau, s, F)$ consists of a finite set of states S, an input alphabet I, an output alphabet O, a starting state s, a set of final states F, and a transition-output function τ from finite subsets of $S \times I^*$ to finite subsets of $S \times O^*$. The transition-output function τ can be also considered as relation $\tau \subseteq S \times I^* \times S \times O^*$.

Returning to the regular transducer $\mathcal{T} = (S, I, O, \tau, s, F)$, for a given word $U \in I^*$, we say that a word $W \in O^*$ is an *output of* $\mathcal{T}$ *for* U if there exists a sequence $(s, U_1, q_1, W_1) \in \tau$, $(s_1, U_2, s_2, W_2) \in \tau$, $\ldots$, $(s_{n-1}, U_n, s_n, W_n) \in \tau$ of state transitions of $\mathcal{T}$, such that $q_n \in F$, $U = U_1 \ldots U_n \in I^*$, and $W = W_1 \ldots W_n \in O^*$. We write $W \in \mathcal{T}(U)$, where $\mathcal{T}(U)$ denotes the set of all outputs of $\mathcal{T}$ for the input word U. For a language $L \subseteq I^*$, we define $\mathcal{T}(L) = \bigcup_{U \in L} \mathcal{T}(U)$. We will often need to refer to the *relation* induced by a transducer $\mathcal{T}$. This relation is a subset of $I^* \times O^*$, and is defined as

$$R_{\mathcal{T}} = \{(U, W) : U \in I^*, W \in \mathcal{T}(U)\}.$$

Relations induced by regular transducers are also called *rational relations* in the literature. For our purposes, we also need to know that rational relations are closed under inverse and union [Yu97].

A regular transducer (S, I, O, τ, s, F) is said to be in the *standard form* if τ is a function from $S \times (I \cup \{\epsilon\})$ to $2^{S \times (O \cup \{\epsilon\})}$. Intuitively. the standard form restricts the input and output of each transition to be only a single letter or ϵ. It is known that any regular transducer is equivalent to a regular transducer in standard form (see [Yu97]).

As discussed in the introduction there are two places where approximations might be called for: the database itself can be an approximate representation of the "real world," or the query issued by the user can be an approximation of the "real query" the user would have submitted, had she known the structure of the database exactly. In both cases we need a mechanism for describing the tolerable "distortions" through which a query or a database can be transformed. Observe that the notion of distortions is similar to the notion of *transformations* in [JMM95]. We describe the set of tolerable distortions by a regular transducer. In this paper we will focus on distorting the query; the treatment where the database is distorted is analogous. Given a query Q and a transducer $\mathcal{T}$, the query Q can be distorted to the query $Q' = \mathcal{T}(Q)$. The set of *$\mathcal{T}$-approximate answers* to a query Q in a graph database DB, given a set of tolerable distortions described by a regular transducer $\mathcal{T}$ is defined as

$$ans_{\mathcal{T}}(Q, DB) = ans(\mathcal{T}(Q), DB).$$

A graph database can be seen as an NFA where the graph nodes are the automaton states and all states are both initial and final. In the "classical" case, computing $ans(Q, DB)$ given an automaton $\mathcal{A}_Q$ for Q and $\mathcal{A}_{DB}$ for the database then essentially amounts to constructing the Cartesian automaton $\mathcal{A}_Q \times \mathcal{A}_{DB}$ and outputting the pair (a, b), if and only if there exists, in the Cartesian automaton, an initial state $(\_, a)$ leading to a final state $(\_, b)$. (see [HSU77, MW95, ABS99]).

We show next that for computing $ans_{\mathcal{T}}(Q, DB)$ we can construct an automaton from the product of $\mathcal{A}_Q$, $\mathcal{T}$, and $\mathcal{A}_{DB}$. The approximate answer can then be read from this automaton, similarly to the "classical" case.

Theorem 1 *Let Q be a query, DB a graph database and $\mathcal{T}$ a distortion transducer. Then a Cartesian transducer $\mathcal{C}$ can be constructed such that $(a, b) \in ans_{\mathcal{T}}(Q, DB)$, if and only if there exists, in $\mathcal{C}$, an initial state $(\_, \_, a)$ leading to a final state $(\_, \_, b)$.*

Proof Sketch. Let $\mathcal{A}_Q = (S_Q, \Delta, \tau_Q, s_{0_Q}, F_Q)$ be an ϵ-free NFA that accepts Q, and let $\mathcal{T} = (S_{\mathcal{T}}, \Delta, \Delta, \tau_{\mathcal{T}}, s_{0_{\mathcal{T}}}, F_{\mathcal{T}})$ be the distortion transducer in standard form. Considering the database DB as another ϵ-free NFA, $\mathcal{A}_{DB} = (S_{DB}, \Delta, \tau_{DB}, S_{DB}, S_{DB})$, we construct the transducer $\mathcal{C} = (S, \Delta, \Delta, \tau, S_0, F)$, where $S = S_Q \times S_{\mathcal{T}} \times S_{DB}$, $S_0 = s_{0_Q} \times s_{0_{\mathcal{T}}} \times S_{0_{DB}}$, $F = F_Q \times F_{\mathcal{T}} \times F_{DB}$, and transition relation τ is defined by, for $(p, q, s) \in S$ and $R_1, R_2 \in \Delta \cup \{\epsilon\}$,

$$\begin{aligned}
\tau \;=\; & \{((p,q,s), R_1, (p',q',s'), R_2) : (p, R_1, p') \in \tau_Q \text{ and } (q, R_1, q', R_2) \in \tau_{\mathcal{T}} \text{ and } (s, R_2, s') \in \tau_{DB}\} \cup \\
& \{((p,q,s), \epsilon, (p,q',s'), R_2) : (q, \epsilon, q', R_2) \in \tau_{\mathcal{T}} \text{ and } (s, R_2, s') \in \tau_{DB}\} \cup \\
& \{((p,q,s), R_1, (p',q',s), \epsilon) : (p, R_1, p') \in \tau_Q \text{ and } (q, R_1, q', \epsilon) \in \tau_{\mathcal{T}}\}
\end{aligned}$$

∎

From the above theorem we now know that the approximate query answering is decidable. Recall that the corresponding problem for the rule based formalism proposed in [JMM95] is undecidable in its full generality.

Ranking the approximate answers

So far we have considered $ans_{\mathcal{T}}(Q, DB)$ as a pure set. The way the distortions are defined, it makes sense to define *distances* between words in the relation $R_{\mathcal{T}}$. The well known *edit distance* is an example of such a distance. Observe that if we rule out transitions of the form ϵ/ϵ from a transducer, since such transitions are useless for any distortion, then each transition in the transducer corresponds to an edit operation. Namely, transitions of the form ϵ/R correspond to insertions, R/ϵ corresponds to deletion, and R/S, where $R \neq S$ corresponds to substitution. Each transition of the above forms is called an *elementary distortion.* It is easy to see that regular transducers can be extended by attaching non-negative *weights* to the transitions. Different insertions, deletions, and substitutions can be given user specified weights in this fashion. Transitions of the form R/R are called *matches* and they can be weighted 0. Given a path p in a transducer, we define the distortion $\delta(p)$ induced by this path as the sum of the weights along this path. Given two words U and W, and a distortion transducer $\mathcal{T}$, the $\mathcal{T}$-distance $d_{\mathcal{T}}(U, W)$, is defined as

$$d_{\mathcal{T}}(U,W) = \begin{cases} inf\{\delta(p) : p \text{ is an accepting path in } \mathcal{T}, in(p) = U, \text{ and } out(p) = W\} \\ \infty \text{ otherwise} \end{cases}$$

where $in(p)$ is the word of input symbols labeling the path p and $out(p)$ is the word of output symbols labeling the path p.

Suppose we have a query Q, a database DB and a distortion transducer $\mathcal{T}$. For each pair (a, b) of database objects in the $\mathcal{T}$-approximate answer of the query Q, we denote with $DB_{a,b}$ the regular language of words labeling the database paths between a and b (cf. [MW95]). Then, in order to rank the pair (a, b) we need to find the closest words $U \in Q$ and $W \in DB_{a,b}$ with respect to the distance $d_{\mathcal{T}}$. Formally, given a query Q, a database DB, and a distortion transducer T, the *rank* is a function from all pairs of database objects to $\mathbb{N} \cup \{\infty\}$ defined by

$$rank(a, b) = inf\{d_{\mathcal{T}}(U, W) : U \in Q, V \in DB_{a,b}\}.$$

Obviously $(a, b) \in ans_{\mathcal{T}}(Q, DB)$ if and only if $rank(a, b) \neq \infty$.

The *ranking problem* for $ans_{\mathcal{T}}(Q, DB)$ is to order the set with respect to the *rank*-function.

Theorem 2 *The set $ans_{\mathcal{T}}(Q, DB)$ can be ranked in time $\mathcal{O}((|\mathcal{A}_Q| \times |DB| \times |\mathcal{T}|)^3)$, where $|\mathcal{A}_Q|$ is the number of states in an ϵ-free automaton for Q, $|DB|$ is the number of database objects, and $|\mathcal{T}|$ is the number of states in the distortion transducer.*

Proof. Consider the Cartesian transducer $\mathcal{C}$ constructed in the proof of Theorem 1. We can view $\mathcal{C}$ as a directed weighted graph. The weight of an edge is a natural number when when the edge is labeled by a transition of the form R/ϵ, ϵ/R, or R/S, where $R \in \Delta$ and $R \neq S$. The weight is 0 for R/R-labelled edges. Then the rank of a pair (a, b) is equal to the shortest path between an initial state $(\_, \_, a)$ and a final state $(\_, \_, b)$. For shortest paths, both Dijskstra's algorithm and Floyd-Warshall algorithm (see e.g. [AHU74]) have the asymptotic worst-case running time mentioned in the claim. ■

Although the running times for both Dijkstra's and Floyd-Warshall algorithms are asymptotically the same, perhaps Dijkstra's algorithm is better suited in our scenario. First, in practice the user might be interested in computing only objects reachable by Q-paths only from a limited number of objects, for example when we have a rooted database graph. In such a case the running time of Dijkstra's algorithm is $\mathcal{O}((|\mathcal{A}_Q| \times |DB| \times |\mathcal{T}|)^2)$, and we don't need to compute the shortest paths between all pairs of objects, as in the Floyd-Warshall algorithm. The second reason has to do with a natural generalization of the approximate answering of a query. Most of the times the user is interested only in the top k-answers. Then Dijkstra's algorithm is the ideal choice: It processes the nodes in the order of their distance from the source. Obviously, we can construct the transducer $\mathcal{C}$ on the fly and stop the execution of the algorithm when the first pair (a, b) of objects, ranked as $k+1$ in $ans_{\mathcal{T}}(Q, DB)$, is produced.

3 Approximate Satisfaction of Data Guides

Data guides were originally introduced in the Lore project as a concise and accurate summary of a given database graph [GW97, N+97]. The data guide is a schema, and we want the database to conform to this schema. However, sometimes we are willing to consider databases that approximately conform to a given data guide.

In such cases we would like to have some quantitative estimation of the approximate conformation of the database to the data guide. Usually, both the database DB and the data guide DG are considered to be edge labelled graphs. The question of whether DB satisfies DG is determined by whether there exists a simulation or bisimulation from DB to DG. If we consider both DB and DG as automata, then simulation corresponds to $L(DB) \subseteq L(DG)$, and bisimulation corresponds to the equality of these languages (see e.g. [ABS99]).

Without loss of generality, we only consider language inclusion (not equality) in this paper. We again suppose that the user specifies a weighted distortion transducer $\mathcal{T}$, through which we can distort the data-guide through allowed elementary distortions and then test if the database DB conforms to the distorted data-guide $\mathcal{T}(DB)$, in other words, if $L(DB) \subseteq L(\mathcal{T}(DG))$.

If the database indeed conforms to the distorted data guide, we are interested in "how far" the database is from the original data guide.

Suppose that indeed $L(DB) \subseteq L(\mathcal{T}(DG))$. Then for each word $W \in L(DB)$ there exists a set of words $U \in L(DG)$, such that $W \in \mathcal{T}(U)$. For each W we want to find the closest such U, and the distance between $L(DG)$ and $L(DB)$ is then the largest among these closest distances (like the diameter of a graph). To formally capture this distance between languages, we need the following definitions.

Given a distortion transducer $\mathcal{T}$, let W be a word in the second column of $R_{\mathcal{T}}$, and L be the set of words appearing in the first column of $R_{\mathcal{T}}$. The $\mathcal{T}$-distance between

L and W is defined as

$$d_{\mathcal{T}}(L, W) = inf\{d_{\mathcal{T}}(U, W) : U \in L\}.$$

Then let L_1 be the set words appearing in the first column of $R_{\mathcal{T}}$, and L_2 those appearing in the second column. The $\mathcal{T}$-distance between L_1 and L_2 is defined as

$$d_{\mathcal{T}}(L_1, L_2) = sup\{d_{\mathcal{T}}(L_1, W) : W \in L_2\}.$$

Now we can say that a database DB *k-satisfies* a data guide DG if

1. $L(DB) \subseteq L(\mathcal{T}(DG))$, and
2. $d_{\mathcal{T}}(\mathcal{T}(DG), L(DB)) \leq k$,

where $\mathcal{T}(DG)$ is the subset of words in $L(DG)$ that appear in the first column of $R_{\mathcal{T}}$.

We are going to prove the following lower bound:

Theorem 3 *Given a data-guide DG, a database DB, a distortion transducer $\mathcal{T}$, and an integer $k \in \mathbb{N}$, the problem of deciding whether or not $d_{\mathcal{T}}(\mathcal{T}(DG), L(DB)) \leq k$, is PSPACE-hard, even if $L(DB) \subseteq L(\mathcal{T}(DG))$.*

For the upper bound of we need the notion of a *distance automata* [Has82]. A distance automaton $\mathcal{A} = (S, \Delta, S_0, \tau, F, \omega)$ is an automaton with positive weights on its transitions. Formally, the weight is a function $\omega : S \times \Delta \times S \to \mathbb{N} \cup \{\infty\}$, such that $\omega(s, R, t) \in \mathbb{N}$ when $(s, R, t) \in \tau$ and $\omega(s, R, t) = \infty$ when $(s, R, t) \notin \tau$. The weight function ω is extended to $\omega : S \times \Delta^* \times S \to \mathbb{N} \cup \{\infty\}$ as follows: for any $s, t \in Q$, $W \in \Delta^*$ and $R \in \Delta$,

1. $\omega(s, \epsilon, t) = \begin{cases} 0 & \text{if } s = t \\ \infty & \text{if } s \neq t \end{cases}$
2. $\omega(s, WR, t) = inf\{\omega(s, W, t') + \omega(t', R, t) : t' \in S\}$.

Then, the distance of a word W accepted by this distance automaton is defined as

$$d(W) \quad = \quad inf\{\omega(s, W, t) : W \in L(\mathcal{A}), s \in S_0 \text{ and } t \in F\}$$

We also need the following definition. Let $\mathcal{A} = (S, \Delta, \tau, S_0, F, \omega)$ be a distance automaton. Then $\mathcal{A}$ is said to have *limited distance* if there is $k \in \mathbb{N}$ such that $k \geq sup\{d(W) : W \in L(\mathcal{A})\}$.

We will show in Theorem 5 that the problem of the k-satisfaction of a data guide DG by database DB through a distortion transducer can be casted to the problem of limitedness in distance automata. The problem of limitedness in distance automata is shown to be decidable [Has82]. This means in turn that our problem of k-satisfaction is decidable. However, the decision procedure of [Has82] is in EXPSPACE and the problem of a tighter lower bound is still open [Has00]. In the next theorem we prove that for the subclass of distance automata with all the states final, the limitedness problem can be decided in PSPACE. As we discussed in the Introduction, this subclass of distance automata corresponds to practical subproblems of k-satisfaction.

Theorem 4 *Given a distance automaton with all states final, i.e.* $\mathcal{A} = (S, \Delta, \tau, S_0, S, \omega)$, *the problem of deciding whether or not the automaton* $\mathcal{A}$ *has limited distance is in PSPACE.*

We now turn to the reduction of the k-satisfaction problem into the problem of limitedness in distance automata. Consider a data-guide DG, a database DB, and a distortion transducer $\mathcal{T}$. In the following we will show a construction for a distance automaton such that the language accepted by it, is exactly the language of the database paths and the cost of each DB path will represent the smallest $\mathcal{T}$-distance of this path from the set of words related through the transduction, and included in the data-guide language.

Let $\mathcal{A}_{DG} = (S_{DG},\, \Delta,\, \delta_{DG},\, S_{0_{DG}},\, F_{DG})$ be an ϵ-free data-guide automaton, $\mathcal{A}_{DB} = (S_{DB},\, \Delta,\, \delta_{DB},\, S_{0_{DB}},\, F_{DB})$ the database automaton, and finally, let $\mathcal{C} = \mathcal{A}_{DG} \times \mathcal{T} \times \mathcal{A}_{DB} = (S,\, \Delta,\, \Delta,\, \tau,\, S_0,\, F)$ be a Cartesian transducer constructed as in the Theorem 1.

Lemma 1 *For* $U \in \mathcal{T}(DG)$ *and* $W \in L(DB)$ *we have that*

$$d_{\mathcal{T}}(U, W) = d_{\mathcal{C}}(U, W)$$

From the transducer $\mathcal{C}$ we will construct another "distance equivalent" transducer $\mathcal{C}'$, with ϵ-free output automaton, and with the same set of states, as follows.

Consider the graph derived by $\mathcal{C}$ consisting of the edges with ϵ output only. Call it $\mathcal{E}_{\mathcal{C}}$. The edges of $\mathcal{E}_{\mathcal{C}}$ will be labeled and weighted with same label and cost as the corresponding transitions in $\mathcal{C}$. We shall use $\epsilon - CLOSURE(s)$, similarly to [HU79], to denote the set of all vertices t such that there is path from s to t in $\mathcal{E}_{\mathcal{C}}$.

Obviously, all the transitions with non-ϵ output of $\mathcal{C}$ will be present in the transducer $\mathcal{C}'$, that we are constructing. Furthermore, we will introduce a transition with output labeled $R \neq \epsilon$ in $\mathcal{C}'$ from a state s to a state t whenever there is a path from vertex s to a vertex v in $\mathcal{E}_{\mathcal{C}}$, and a transition with R labeled output in $\mathcal{C}$, from the state v to the state t. Formally, if $\mathcal{C} = (S,\, \Delta,\, \Delta,\, \tau,\, S_0,\, F)$, then $\mathcal{C}' = (S,\, \Delta,\, \Delta,\, \tau',\, S_0,\, F')$, where $F' = F \cup \{s : s \in S_0 \text{ and } \epsilon - CLOSURE(S_0) \cap F \neq \emptyset\}$ and

$$\begin{aligned} \tau' \;=\; & \{(s, R', t, R) : (v, R', t, R) \in \tau \text{ and } R \neq \epsilon\} \cup \\ & \{(s, W, t, R) : \exists v \in \epsilon - CLOSURE(s) \text{ such that,} \\ & \;\text{there is a transition } (v, R', t, R) \in \tau, \text{ where } R \neq \epsilon\} \end{aligned}$$

where W will be a word labeling the(a) cheapest path from vertex s to vertex v in $\mathcal{E}_{\mathcal{C}}$. Also, the cost of a new transition (s, W, t, R) will be the cost of the(a) corresponding cheapest path from s to v in $\mathcal{E}_{\mathcal{C}}$, plus the cost of the(a) cheapest transition with output labeled by R, from state v to state t in $\mathcal{C}$.

Lemma 2 *Let* $\mathcal{C}'$ *be an* ϵ*-free output Cartesian transducer constructed as above. Then the following are true.*

1. $out(\mathcal{C}') = out(\mathcal{C})$

2. $in(\mathcal{C}') \subseteq in(\mathcal{C})$

3. $R_{\mathcal{C}'} \subseteq R_{\mathcal{C}}$

Lemma 3 *Let $W \in out(\mathcal{C}) = out(\mathcal{C}')$. Then $d_{\mathcal{C}}(in(\mathcal{C}), W) = d_{\mathcal{C}'}(in(\mathcal{C}'), W)$*

If we now eliminate the input from the transitions in $\mathcal{C}'$ we obtain an ϵ-free distance automaton $\mathcal{A} = (S, \Delta, \tau'', S_0, F, \omega)$, where

$$\tau'' = \{(p, R, q) : (p, W, R, q) \in \tau' \text{ for some } W\},$$

and the weight ω of a transition in $\mathcal{A}$ is given by the cost of the(a) corresponding cheapest transition in the transducer $\mathcal{C}'$. Now, we can state the following theorem.

Theorem 5 *Let DG be a data-guide, DB a database, and $\mathcal{T}$ a transducer of allowed distortions. Compute the ϵ-free output Cartesian transducer $\mathcal{C}'$ and consider the distance automaton $\mathcal{A}$ constructed as previously. Then, $dist(\mathcal{A}) = d_{\mathcal{T}}(\mathcal{T}(DG), L(DB))$.*

References

[Abi97] S. Abiteboul. Querying Semistructured Data. *Proc. of ICDT* 1997 pp. 1–18.

[ABS99] S. Abiteboul, P. Buneman and D. Suciu. *Data on the Web : From Relations to Semistructured Data and Xml.* Morgan Kaufmann, 1999.

[AHU74] A. Aho, J. E. Hopcroft and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley 1974.

[B+97] P. Buneman, S. B. Davidson, M. F. Fernandez and D. Suciu. Adding Structure to Unstructured Data. *Proc. of ICDT* 1997, pp. 336–350.

[C+99] 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.

[C+00b] D. Calvanese, G. Giacomo, M. Lenzerini and M. Y. Vardi. Answering Regular Path Queries Using Views. *Proc. of ICDE* 2000, pp. 389–398

[CP97] C. Choffrut and G. Pighizzini. Distances Between Languages and Reflexivity of Relations. *Proc. of MFCS* 1997, pp. 199–208

[FS98] M. F. Fernadez and D. Suciu. Optimizing Regular path Expressions Using Graph Schemas *Proc. of ICDE* 1998, pp. 14–23.

[GW97] R. Goldman, J. Widom DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. *Proc. of VLDB* 1997, pp. 436–445.

[GT00] G. Grahne and A. Thomo. An Optimization Technique for Answering Regular Path Queries *Informal Proc. of WebDB* 2000 pp. 99–104.

[GT01] G. Grahne and A. Thomo. Algebraic rewritings for optimizing regular path queries. *ICDT* 2001, pp. 303–315

[Has82] K. Hashiguchi. Limitedness Theorem on Finite Automata with Distance Functions. *J. Comp. Syst. Sci.* 24, 1982 pp. 233–244.

[Has00] K. Hashiguchi. New upper bounds to the limitedness of distance automata. *Theoretical Computer Science* 233(1-2), 2000 pp. 19–32.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley 1979.

[HRS76] H. B. Hunt III, D. J. Rosenkrantz, and T. G. Szymanski, On the Equivalence, Containment, and Covering Problems for the Regular and Context-Free Languages. *J. Comp. Syst. Sci.* 12(2) 1976, pp. 222–268

[HSU77] H. B. Hunt III, T. G. Szymanski, and J. D. Ullman. Operations on sparse relations. *Comm. ACM* 20(3), 1977, pp. 171–176

[JMM95] H. V. Jagadish, Alberto O. Mendelzon, Tova Milo. Similarity-Based Queries. *Proc. PODS* 1995 pp. 36–45.

[MW95] A. O. Mendelzon and P. T. Wood, Finding Regular Simple Paths in Graph Databases. *SIAM J. Comp. 24:6,* (December 1995).

[MMM97] 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.

[N+97] S. Nestorov, J. D. Ullman, J. L. Wiener, S. S. Chawathe. Representative Objects: Concise Representations of Semistructured, Hierarchial Data. *Proc. of ICDE*, 1997, pp. 79–90.

[Yu97] S. Yu. Reqular Languages. In: *Handbook of Formal Languages.* G. Rozenberg and A. Salomaa (Eds.). Springer Verlag 1997, pp. 41–110