

# Indexes and path constraints in semistructured data \*

Yves André

Anne-Cécile Caron

Denis Debarbieux

Yves Roos

Mostrare project, RU INRIA Futurs

LIFL, UMR 8022 CNRS, Université de Lille I.

E-mail: {andre, caronc, debarbie, yroos}@lifl.fr

## Abstract

In this paper, we study semistructured data and indexes preserving inclusion constraints. A semistructured datum is modelled by multi-rooted edge-labeled directed graphs. We consider regular path queries and inclusion constraints over these data. These constraints are binary relations over regular path expressions  $q$  and  $r$ , and are interpreted on a datum as “for this datum, the answer to query  $q$  is included in the answer to query  $r$ ”. We study how to represent inclusion constraints that are common to several data. Our work is based on two existing indexes: dataguide and 1-index. Given a set of data  $S$ , we extract from the dataguide of  $S$  a finite set  $C(S)$  of (finite) inclusion constraints such that an inclusion constraint is satisfied by a datum of  $S$  if and only if it is implied by  $C(S)$ . We use 1-index which are covering indexes preserving inclusion constraints. Experiments compare the different ways of using the 1-index to index a set of data.

keywords: (set of) semistructured data, regular path expressions, inclusion constraints, indexes, XML

## 1 Introduction

A semistructured datum is a datum which structure is unknown but can be inferred from the datum. An XML document (with or without references) and HTML pages (with hyper-links) are examples of semistructured datum. So a semistructured datum is heterogeneous, complex and does not own a fixed schema like relational data. In this article, we see semistructured data as multi-rooted edge-labeled directed graphs. A presentation of this model and an overview of works done in this context can be found in [1].

There are many ways to query a semistructured datum. We consider queries based on the paths appearing in the datum:

\*This research was partially supported by: “CPER 2000-2006, Contrat de Plan état - région Nord/Pas-de-Calais: programme TAC, projet COCOA” and “ACI masse de données TRALALA, FNS”

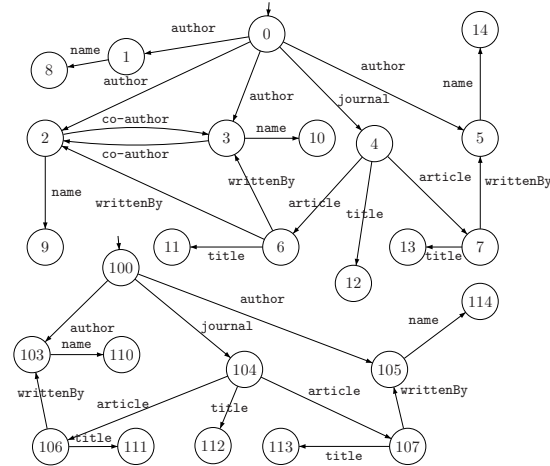


Figure 1. Example of semistructured datum

we use the labels of the datum to form paths and to navigate in the datum. A *regular query* is a regular expression on the alphabet of labels appearing in the datum. The result of the regular query  $q$  is the set of nodes reached from the root(s) by the paths labeled by any word  $u$  of  $q$ . By extension, this word  $u$  is called *label path*. If  $S$  is a set of data then querying  $S$  consists in querying each datum of  $S$  and computing the union of the solutions. For example, figure 1 shows a set  $S$  with two data. `author`, `journal.article`, are label paths of the set  $S$ . The regular expression `author.co-author*` is a regular query whose result on first datum is  $\{1, 2, 3, 5\}$  and whose result on the second one is  $\{103, 105\}$ . So the result of  $q$  on  $S$  is the set  $\{1, 2, 3, 5, 103, 105\}$ .

Let us note that there is no difference, in the query point of view, between a set  $S$  and the datum  $\langle N_S, R_S, T_S \rangle$  defined by: the set of nodes  $N_S$  is the union of the sets of nodes of the datum in  $S$  (we suppose, without loss of generality, that two different data have no common nodes), the set of roots  $R_S$  is the union of all the roots of the data of  $S$ , and the set of edges  $T_S$  is the union of the set of edges of

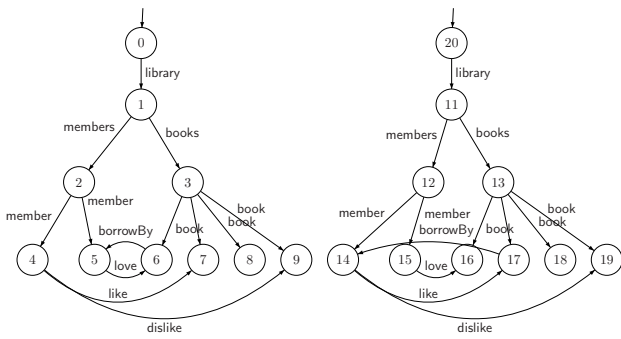


Figure 2. Library database

the data of  $S$ .

Some integrity constraints found in object-oriented databases and also common in semistructured data can be expressed with path constraints. Here, we consider path inclusions. A path inclusion constraint is written  $p \preceq q$  where  $p$  and  $q$  are regular queries, and means that the set of nodes result of  $p$  is included in the set of nodes result of  $q$ . Continuing the example, since the result of `journal.article.writtenBy` is  $\{2, 3, 103, 105\}$  and the result of `author` is  $\{1, 2, 3, 103, 105\}$  the path inclusion constraint `journal.article.writtenBy`  $\preceq$  `author` is satisfied by any datum of  $S$ . These constraints have been introduced by S. Abiteboul and V. Vianu in [2] and some works on path constraints can be found, for instance, in [4], [6] [5], [12] or [3].

Indexing **one datum** is a well known problem: see for example the dataguides [11], perfect indexes [15], 1-indexes [14], A(k)-indexes [13], D(k)-indexes [7]. In this paper we study how to index a set of data  $S$  (figure 2) such that:

1. The index is a covering index i.e. any query can be evaluated from the index alone, without consulting the set of data.
2. The index is a multi-rooted graph which satisfies a path inclusion constraint  $p \preceq q$  if and only if any datum  $D$  of the set  $S$  models the constraint  $p \preceq q$ .
3. The index can be efficiently computed.

As far as we know, item 2 has never been studied. Items 1 and 3 are well known problems in semistructured database community and we propose to use index for one datum in order to index a set of data. There are two possibilities to do this transformation. One can compute:

1. The union of the indexes of each datum denoted by  $\text{U-index}(S)$
2. The index of the set  $S$  considered as one datum, denoted by  $\text{index}(S)$

In this paper we use dataguide [11] to index a set of data. In particular we propose a variant of dataguide which models the word inclusion constraints satisfied by any datum and we can extract from it a set of constraints  $\mathcal{C}(S)$  such that  $\mathcal{C}(S) \models p \preceq q$  if and only if  $\forall D \in S D \models p \preceq q$ . In this case the extraction algorithm produces the same result for  $\text{U-index}(S)$  as for  $\text{index}(S)$ .

In a second part we use the 1-index [14]. We prove that, for any set  $S$  of data,  $\text{U-index}(S)$  and  $\text{1-index}(S)$  model an inclusion constraint if and only if this constraint is satisfied by any datum of  $S$ . Moreover, we prove that  $\text{1-index}(S)$  is always smaller than  $\text{U-index}(S)$ . We made some experiments in order to check whether the size of  $\text{1-index}(S)$  is significantly smaller than the size of  $\text{U-index}(S)$ . On the other hand, the  $\text{U-index}(S)$  seems to be more adapted when one does not want to query  $S$  but only a subset  $S'$  of  $S$ . Other experiments study the link between the fraction size of  $S'$  w.r.t. size of  $S$  and the time save (lost) by querying  $S'$  with  $\text{1-index}(S)$  rather than  $\text{U-index}(S)$ .

## 2 Dataguide

R. Goldman and J. Widom have introduced the notion of (strong) dataguide in [11] in order to index the paths of a datum. The strong dataguide  $SD(D)$  of a datum  $D$  is a datum such that

1. Every label path of  $D$  occurs exactly once in  $SD(D)$ .
2. Every label path of  $SD(D)$  is a label path of  $D$ .
3. For any label path  $p$ ,  $L_D(p) = L_{SD}(p)$  where  $L_D(p)$  denotes the set of the label paths in  $D$  that have the same result as  $p$  (i.e.  $L_D(p) = \{p' \mid \text{result}_D(p) = \text{result}_D(p')\}$ ).

A datum  $D$  can be seen as a word automaton considering each node as a terminal state and each root as an initial state. It follows, from automata theory, that the strong dataguide  $SD(D)$  is the deterministic automaton obtained by the well known subset construction applied on datum  $D$ . Then R. Goldman and J. Widom directly obtain that  $SD(D)$  is a covering index of  $D$  and its number of nodes can be exponentially bigger than the number of nodes of datum  $D$ .

Figure 3 shows a datum that satisfies  $a \preceq b$  whereas the dataguide (on right hand side of the figure and without the dot lines) does not satisfy this constraint. It follows that the strong dataguide does not preserve path inclusion constraints. Nevertheless, this constraint is not completely lost since  $\text{result}_{SD(D)}(a) = \{1\}$ ,  $\text{result}_{SD(D)}(b) = \{1, 2\}$  and  $\{1\}$  is included in  $\{1, 2\}$ .

In spite of those two problems (size, constraints), the strong dataguide has good properties allowing to obtain all inclusion constraints which are present in a datum :

On one hand, thanks to the subset construction, any node  $s$  in a strong dataguide  $SD(D)$  of a datum  $D$  can be seen as a set  $s(D)$  of nodes of  $D$ . Let  $u$  and  $v$  be two words such that  $\text{result}_{SD(D)}(u) = \{s\}$ ,  $\text{result}_{SD(D)}(v) = \{s'\}$  and  $s(D)$  is included in  $s'(D)$  then  $D \models u \preceq v$ . In order to capture any word inclusion present in  $D$ , we define the *saturated strong dataguide* using a saturation operator quite similar to the saturation operator defined in [10]: let  $D$  be a datum and  $SD(D) = \langle N, \{r\}, T \rangle$  its strong dataguide, the saturated strong dataguide  $SD_s(D)$  is the datum  $\langle N, I', T' \rangle$  where

- $I' = \{n \in N \mid n(D) \subseteq \text{result}_{SD}(\epsilon)\}$ ,
- $T' = \{(n, x, n') \mid \exists n'' \in N, (n, x, n'') \in T \wedge n'(D) \subseteq n''(D)\}$ .

Then for any label paths  $u$  and  $v$ , and for any set of data  $S$ , we get :

$$SD_s(S) \models u \preceq v \text{ if and only if } \forall D \in S \ D \models u \preceq v$$

Moreover, we have proved in [9] that one can extract from the strong dataguide  $SD(D)$  of any datum  $D$  a *finite* set of *finite* path inclusions  $\mathcal{C}(D)$  such that  $D \models p \preceq q$  if and only if  $\mathcal{C}(D) \models p \preceq q$ . It follows that if we consider now a set  $S$  of data, one can extract from the strong dataguide  $SD(S)$  of  $S$  a finite set  $\mathcal{C}(S)$  of constraints such that

$$\mathcal{C}(S) \models p \preceq q \text{ if and only if } \forall D \in S \ D \models p \preceq q$$

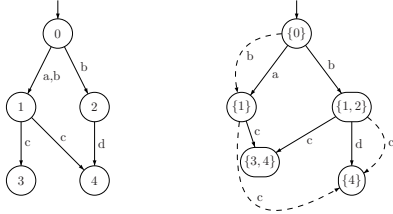


Figure 3. Saturated strong dataguide

**Example 1** Figure 3 shows a datum  $D$  and its saturated strong dataguide  $SD_s(D)$ . For instance, since  $\{4\}$  is included in  $\{3,4\}$  and there exists a transition  $(\{1,2\}, c, \{3,4\})$ , we add a transition (dot line)  $(\{1,2\}, c, \{4\})$  in  $SD_s(D)$ . As, in datum  $D$ , label path  $b$  reaches nodes 1 and 2 and label path  $a$  reaches only node 1, we have  $D \models a \preceq b$ . Since transition  $(\{0\}, b, \{1\})$  is added in the saturated strong dataguide,

### 3 1-index

Strong dataguides are indexes whose size could be exponentially bigger than the size of the datum. T. Milo and

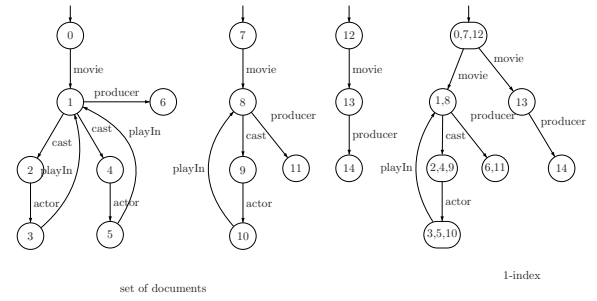


Figure 4. Set of data and 1-index

D. Suciú propose in [14] another indexation method in order to limit the size of the index. This method merges nodes that are equivalent (i.e. not distinguishable from the query point of view). If  $n$  is a node of a datum  $D$ , let  $L_n(D)$  be the set of the label paths from some root node to  $n$ . Then, they define an equivalence relation on nodes by:

$$n \equiv n' \text{ if } L_n(D) = L_{n'}(D)$$

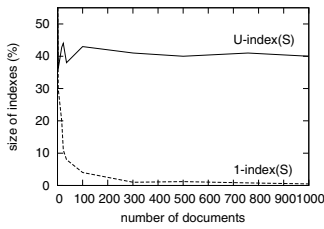
and the index is the quotient of the datum by this relation. It is obvious that this index is covering and is smaller than the initial datum. But the construction of the index is very expensive since computing the equivalence classes is a PSPACE-complete problem ([17]). To tackle this construction cost, T. Milo and D. Suciú consider refinements i.e. equivalence relations  $\approx$  on nodes s.t.  $n \approx n'$  implies that  $n \equiv n'$ .

Two nodes  $n$  and  $n'$  are backward bisimilar ( $n \equiv_b n'$ ) if there exists a backward bisimulation  $\sim$  s.t.  $n \sim n'$ . Based on R. Paige and T.E. Tarjan algorithm ([16]) one can compute the equivalence relation  $\equiv_b$  for labeled graphs in  $O(m \cdot \log(m))$  (where  $m$  in the number of edges).

1-index(D) is a multi-rooted edge labeled graph defined as follows: its nodes are equivalence classes  $[n]$  for  $\equiv_b$ ; for each edge  $(n, x, n')$  in  $D$  there exists an edge  $([n], x, [n'])$  in 1-index(D). The roots are  $[r]$  for each root  $r$  in  $D$ .

As  $\equiv_b$  is a refinement of  $\equiv$  for any node  $n$  in the datum,  $L_n(D) = L_{[n]}(1\text{-index}(D))$ . T. Milo and D. Suciú have established in [14] that the 1-index is a covering index, and  $D$  and 1-index(D) have the same strong dataguide. We deduce from this remark that  $\mathcal{C}(S)$  the set of constraints extracted from  $S$  is equal to  $\mathcal{C}(1\text{-index}(S))$  the set of constraints extracted from 1-index(S). This proves that the following properties are equivalent:

1.  $\forall D \in S, D \models p \preceq q$
2.  $1\text{-index}(S) \models p \preceq q$
3.  $U\text{-index}(S) \models p \preceq q$



**Figure 5. Experiments on the movies base**

We can compare  $1\text{-index}(S)$  and  $U\text{-index}(S)$  on two criteria: their sizes and the time used to answer queries. For this second criterion, another interesting problem is to evaluate a query on a subset of data.

Concerning the size, we can prove that for any set  $S$ ,  $\text{size}(S) \geq \text{size}(U\text{-index}(S)) \geq \text{size}(1\text{-index}(S))$ . Indeed two nodes merged in  $U\text{-index}(S)$  are also merged in  $1\text{-index}(S)$ . As we consider data which belong to the same set, we can imagine that they are similar and that many nodes of different data are backward bisimilar. In this case, the size of  $1\text{-index}(S)$  should be significantly smaller than the size of  $U\text{-index}(S)$ . This is confirmed by the experiments presented figure 5, where x-axis corresponds to the size of  $S$  (i.e. the number  $n$  of data) and y-axis corresponds to percent of the data set size. Let us note that the authors of [13] have established by experiments that the size of the 1-index should be 45% of the size of the datum.

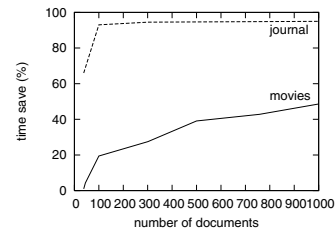
Concerning the time, there exists a  $O(|D| \times |q|)$  algorithm for answering to a query  $q$  on a document  $D$ . So if we compare the time complexity using 1-index and U-index, we will get the same conclusion than for the space complexity. But what about querying a subset  $S'$  of  $S$ ? In this case, we compare the two following methods: first, query the  $1\text{-index}(S)$  and then project the result on  $S'$ ; second, query a sub part of  $U\text{-index}(S)$  denoted  $U\text{-index}(S')$ .

We know that the time needed to answer  $q$  using  $1\text{-index}(S)$  is independent of  $S'$  since querying  $1\text{-index}(S)$  does not depend on  $S'$  and projecting the result on  $S'$  need to map any node of  $\text{result}_{1\text{-index}(S)}(q)$ . On the opposite side, the algorithm which uses  $U\text{-index}(S)$  depends on  $S'$ : if  $S'$  is reduced to one datum, then  $U\text{-index}(S')$  is always faster but if  $S'$  is equal to  $S$  then  $1\text{-index}(S)$  is faster. So the answer to the question depends on the fraction:  $\frac{\text{size of } S'}{\text{size of } S}$ .

This leads to the following experiment: for a given query  $q$ , a given set of datum  $S = \{D_i, 1 \leq i \leq n\}$ , we construct all the sets  $S_j = \{D_i, 1 \leq i \leq j\}$  and we query  $S$  and  $S_j$  with  $q$  for any  $j$ .

The result of the experiment is presented figure 6 where the x-axis corresponds to the number  $n$  of data and the y-axis corresponds to the time save using  $1\text{-index}(S)$  instead of using  $U\text{-index}(S)$ .

Let us present the data sets used for our experiments. Those



**Figure 6. Time save using 1-index(S)**

experiments have been done with QRIC [8].

**Data set 1** We consider a set of XML data (with references) about the movies produced in 1966 (figure 4 presents a part of the base). This base is produced from The International Movie Database. As seen in figure 4 the data look similar but have some differences. In particular the number of actors is different and sometimes there is no actor (the field is missing) and it has some consequences on the 1-index. For example nodes 8 and 13 cannot be merged since the label path `movie.cast.author.playIn` reaches node 8 and not node 13.

Nevertheless figure 5 shows that the size of the U-index of any movie datum is 41% of the size of the datum whereas the size of the 1-index of the set is smaller. Moreover it seems that, if the set is big enough, the 1-index becomes constant. Indeed when all particular cases are stored in the 1-index, this one does not grow up anymore.

Figure 6 (line movies) shows that we can index all the data of  $S$  together without loss of efficiency even if we query a subset  $S'$  of  $S$ .

**Data set 2** We made some experiments on set of journal data (see figure 1). We use ToXgene [18] to generate a random set of data such that two different data have very similar 1-indexes. So we obtain very good results as shown in figure 6 (line journal).

**Data set 3** Figure 2 shows another kind of data: a set in which each datum represents a library. Each member of this library has some opinion (love, like, dislike, notFinish) about some books and the books are borrowed (or reserved) by zero or one member. As before we use ToXgene [18] to generate a set of data.

Our experiments show that the 1-index is about 58% the size of the datum (figure 7). Unfortunately, this size is not much smaller than the size of U-index. Indeed we establish, by experiments, that  $\text{size}(U\text{-index}(S)) \sim \text{size}(1\text{-index}(S))$  (figure 7). Since there are many different possible cycles in those data (e.g. any path cycle which belongs to  $L(((\text{love} + \text{like} + \text{dislike} + \text{notFinish}).(\text{borrowBy} + \text{reserveBy}))^*)$ ), there are

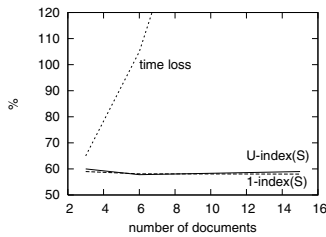


Figure 7. Experiments of the library base

many nodes which have different in-going languages and cannot be merged in the 1-index. For instance, one datum of figure 2 has a cycle based on the path `borrowBy.love` whereas the other datum has a cycle based on the path `borrowBy.like`.

Let  $S_j$  be a strict subset of  $S$ . Our experiments show that, since  $\text{size}(\text{U-index}(S_j))$  is always smaller than  $1\text{-index}(S)$ , it is always faster to query  $S_j$  by separately indexing each datum (figure 7).

## 4 Conclusion

In this paper we study the connection between index and path inclusion constraints. More precisely we want, from a set of data  $S$ , to compute efficiently a covering index which preserves the path inclusion constraints modelled by any datum of  $S$ .

As we can extract from the dataguide a finite set of constraints  $\mathcal{C}(S)$  s.t.  $\mathcal{C}(S)$  models  $p \preceq q$  iff  $p \preceq q$  is modelled by  $S$ , we prove that the 1-index satisfies our criterion. Moreover, we already know that the size of  $\text{U-index}(S)$  is always bigger than the size of  $1\text{-index}(S)$ . Our experiments show that we can save a lot of space (and time) when adding a new element to  $S$  does not change the  $1\text{-index}(S)$ . But even if we consider similar data,  $1\text{-index}(S)$  and  $\text{U-index}(S)$  may look similar and may have the same size. In this case, it is better to use  $\text{U-index}$  to query a subset of  $S$ .

**Future work** There already exist algorithms ([19]) to update the 1-index (adding or deleting nodes). Our future work will focus on this topic: how can we update the 1-index when we add (remove) one or more data?

**Acknowledgement** We would like to thank S. Tison for her usefull comments on this paper and J. Luchier for his help in developing the Qric software.

## References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web*. Morgan Kaufmann Publishers, 2000.

[2] S. Abiteboul and V. Vianu. Regular path queries with constraints. In *Proc. of ACM Symposium on Principles of Database Systems*, 1997.

[3] N. Alechina, S. Demri, and M. de Rijke. A modal perspective on path constraints. *Journal of Logic and Computation*, 13(6):939 – 956, 2003.

[4] Y. Andre, A. Caron, D. Debarbieux, Y. Roos, and S. Tison. Extraction and implication of path constraints. In *Proceedings of the 29th Symposium on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, pages 863–875, 2004.

[5] N. Bidoit, S. Cerrito, and V. Thion. A first step towards modelling semistructured data in hybrid modal logic. *Journal of Non Classical Logics*, 2004.

[6] P. Buneman, W. Fan, and S. Weinstein. Path constraints in semistructured databases. *Journal of Computer and System Sciences*, 61(2), 2000.

[7] Q. Chen, A. Lim, and K. W. Ong. D(k)-index: an adaptive structural summary for graph-structured data. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 134–144. ACM Press, 2003.

[8] D. Debarbieux and J. Luchier. Qric: Query rewriting with inclusion constraints. [www.lifl.fr/~debarbie/QRIC/](http://www.lifl.fr/~debarbie/QRIC/), 2004.

[9] D. Debarbieux, Y. Roos, and S. Tison. Models of path constraints. In *Proceedings of the 10th Mons Theoretical Computer Science Days*, 2004.

[10] F. Denis, A. Lemay, and A. Terlutte. Residual finite state automata. In *Lecture Notes in Computer Science 2001*, pages 144–157. 18th Annual Symposium on Theoretical Aspects of Computer Science, 2001.

[11] R. Goldman and J. Widom. Dataguides : Enabling query formulation and optimization in semistructured databases. In *Proc. of International Conf. on Very Large Data Bases*, pages 436–445, 1997.

[12] G. Grahne and A. Thomo. Query containment and rewriting using views for regular path queries under constraints. In *proceedings of PODS'03*, pages 111–122. Symposium on Principles of Database Systems, ACM, 2003.

[13] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for efficient indexing of paths in graph structured data. In *ICDE*, 2002.

[14] T. Milo and D. Suciu. Index structures for path expressions. *Lecture Notes in Computer Science*, 1540:277–295, 1999.

[15] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting schema from semistructured data. In *Proc. of ACM SIGMOD International Conference on Management of Data*, June 1998.

[16] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987.

[17] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *STOC*, pages 1–9, 1973.

[18] Toxgene - the tox xml data generator. [www.cs.toronto.edu/tox/toxgene/](http://www.cs.toronto.edu/tox/toxgene/), 2002.

[19] K. Yi, H. He, I. Stanoi, and J. Yang. Incremental maintenance of xml structural indexes. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 491–502. ACM Press, 2004.