Inferring Regular Expressions with Interleaving from XML Data

Xiaolan Zhang[1,2], Yeting Li[1,2], Fei Tian[3], Fanlin Cui[1,2], Chunmei Dong[1,2], and Haiming Chen[1(✉)]

[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
{zhangxl,liyt,cuifl,dongcm,chm}@ios.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China
[3] University of Science and Technology of China, Hefei, China
tf4811@mail.ustc.edu.cn

Abstract. Document Type Definition (DTD) and XML Schema Definition (XSD) are two popular schema languages for XML. However, many XML documents in practice are not accompanied by a schema, or by a valid schema. Therefore, it is essential to devise efficient algorithms for schema learning. Schema learning can be reduced to the inference of restricted regular expressions. In this paper, we first propose a new subclass of restricted regular expressions called *Various CHAin Regular Expression with Interleaving (VCHARE)*. Then based on single occurrence automaton (SOA) and maximum independent set (MIS), we introduce an inference algorithm *GenVCHARE*. The algorithm has been proved to infer a descriptive generalized *VCHARE* from a set of given sample. Finally, we conduct a series of experiments based on our data set crawled from the Web. The experimental results show that *VCHARE* can cover more content models than other existing subclasses of regular expressions. And, based on the data sets of *DBLP*, regular expressions inferred by *GenVCHARE* are more accurate and concise compared with other existing methods.

1 Introduction

Document Type Definition (DTD) and XML Schema Definition (XSD) are two popular schema languages for XML recommended by World Wide Web Consortium (W3C) [31]. The presence of a schema has numerous advantages such as data processing, automatic data integration, static analysis of transformations and so on [2,11,20,22–24,28]. Besides, the existence of schemas is necessary when integrating (meta) data through schema matching [30] and in the area of generic model management [3,26]. However, many XML documents are not accompanied by a (or valid) schema in practice. A survey [19] shows that XML

H. Chen—Work supported by the National Natural Science Foundation of China under Grant No. 61472405.

Y. Cai et al. (Eds.): APWeb-WAIM 2018, LNCS 10988, pp. 44–52, 2018.
https://doi.org/10.1007/978-3-319-96893-3_4

documents on the Web which have schema definitions only account for 24.8% in 2013, of which the proportion of valid schemas is only about 8.9%. Therefore, it is essential to devise algorithms for schema inference. And schema inference can be reduced to learning restricted regular expressions from a set of given sample [6,8,16].

Gold [18] proposed a classical language learning model (*learning in the limit or explanatory learning*) and pointed out that the class of regular expressions cannot be learnable from positive examples only. Furthermore, Bex et al. proved in [4] that even the class of deterministic regular expressions is too rich to be learnable from positive data. Consequently, researchers have turned to study the restricted subclasses of regular expressions [27].

The popular existing subclasses of regular expressions used in XML such as SORE [6], CHARE (Simplified CHARE) [6], eSimplified CHARE [12], Simple regular expression (CHARE) [5], eCHARE [25] were renamed as in the brackets and analyzed together in [21]. These subclasses are all based on standard regular expressions. In data-centric applications using XML, there may be no order constraint among siblings [1]. However, the relative order within siblings may be still important. In [9], Ciucanu and Staworko proposed two schema formalisms for unordered XML: *disjunctive multiplicity expressions (DME)* and *disjunction-free multiplicity expressions (ME)* where the relative order among siblings was ignored. These two formalisms do not support the concatenation within siblings. For example, $E_1 = (a|b)^+\&c$ is a DME and $E_2 = a\&b^*\&c^?$ is an ME. But $E_3 = (a^+b^?)\&c^*$ does not satisfy both two formalisms. Peng and Chen in [29] also focused on the unordered relation among siblings and proposed *SIRE*. *SIRE* supports the concatenation operation within siblings. Therefore E_3 is a $SIRE$. However, *SIRE* does not support union operation. In [17], Ghelli et al. proposed a restricted subclass defined by grammar $T ::= \varepsilon|a^{[m,n]}|T+T|T\cdot T|T\&T$ where $m \in N\backslash\{0\}$ and $n \in N\backslash\{0\} \cup \{*\}$. For this subclass, counters (repetition operation) can only occur as a constraint for terminal symbols of strings in $L(T)$. For example, $E_4 = a^?(b|c|d)^*$ is not allowed.

In this paper, we focus on learning a restricted deterministic regular expression considering interleaving from a set of given positive examples. We propose a new subclass named as *Various CHAin Regular Expression with Interleaving (VCHARE)*. *VCHARE* supports union, concatenation and interleaving operators together. For example, $E_5 = a^*\&b^+\&c^?$ and $E_6 = (a|b^?)(c^*d^?|e^*)^+$ are both *VCHAREs*.

As for learning algorithms for XML data, Bex et al. [6,7] proposed two inference algorithms *RWR* and *CRX* for SOREs and its Simplified CHAREs, respectively. Freydenberger and Kötzing [13] proposed another two inference algorithms $Soa2Chare$ and $Soa2Sore$ based on *Single Occurrence Automaton* (SOA) for Simplified CHAREs and SOREs, respectively. These two algorithms can infer descriptive generalized regular expressions (explained below) while *RWR* and *CRX* can not. Ciucanu and Staworko introduced an algorithms for *DME* based on *max clique* [9]. Peng and Chen [29] proposed an approximation algorithm and heuristic solution to infer a descriptive generalized $SIRE$.

The concept of *descriptive generalization* [14], is different from Gold-style language learning. Gold-style learners are required to infer an exact description for the target language in a class. But descriptive generalization views the hypothesis space and the space of target language as distinct. Here is a formal explanation. For a class $\mathcal{D}$ of language representation mechanisms (e.g., a class of automata, regular expressions, or grammars), a representation $\alpha \in \mathcal{D}$ is called $\mathcal{D}$-descriptive for a set of given sample S if the language of α is an inclusion-minimal generalization of S. It means that there is no $\beta \in \mathcal{D}$ such that $S \subseteq \mathcal{L}(\beta) \subset \mathcal{L}(\alpha)$.

In present paper, the inference algorithm (*GenVCHARE*) is also based on the concept of descriptive generalization which aims to infer descriptive generalized *VCHAREs* for a set of given sample S. The main idea of *GenVCHARE* is based on SOA and *Maximum Independent Set* (MIS). We first construct an SOA for S. Then replace each non-trival strongly connected component (NTSCC) by the return value of *RepairRE()* as one new node. Next, assign each node a level number. Finally, all nodes of each level will be converted to one or more chain factors.

The main contributions of this paper are listed as follows.

- We propose a subclass of restricted regular expressions named as *Various CHAin Regular Expression with Interleaving (VCHARE).*
- We design an inference algorithm *GenVCHARE* to infer descriptive generalized *VCHAREs.*
- We analyze the coverage proportion of *VCHARE* compared with other subclasses based on the real-world data set. Based on the data sets (*DBLP*), we compare the inferred results with other inferrence methods. The experimental results shows that regular expressions inferred by *GenVCHARE* are more accurate.

This paper is organized as follows. In Sect. 2 introduces some basic definitions. Section 3 is the inference algorithm *GenVCHARE.* Section 4 gives the experiments. Conclusions are drawn in Sect. 5.

2 Preliminaries

Definition 1. ***Regular Expression with Interleaving.*** *Let Σ be a finite alphabet. Σ^* is the set of all strings over Σ. A regular expression with interleaving over Σ is inductively defined as follows: ε or $a \in \Sigma$ is a regular expression where $a \in \Sigma$. For any regular expressions E_1 and E_2, the disjunction $E_1|E_2$, the concatenation $E_1 \cdot E_2$, the interleaving $E_1 \& E_2$, or the Kleene-Star E_1^* is also a regular expression. The language generated by E is defined as follows: $L(\emptyset) = \emptyset$; $L(\varepsilon) = \{\varepsilon\}$; $L(a) = \{a\}$; $L(E_1^*) = L(E)^*$; $L(E_1E_2) = L(E_1)L(E_2)$; $L(E_1|E_2) = L(E_1) \cup L(E_2)$; $L(E_1 \& E_2) = L(E_1E_2) \cup L(E_2E_1)$. $E^?$ and E^+ are used as abbreviations of $E + \epsilon$ and EE^*, respectively.*

In the specification of XSD, the interleaving operator is used in the form of $a_1^{c_1} \& a_2^{c_2} \& \cdots \& a_n^{c_n}$ where $a_i \in \Sigma$ and $c_i \in \{1, ?, +, *\}$. For $a, b \in \Sigma$, $x, y \in \Sigma^*$, we have $a \& \epsilon = \epsilon \& a = a$ and $ax \& by = a(x \& by) \cup b(ax \& y)$.

Let S be the set of given sample. $POR(S)$ is the set of all partial order relations of each string in S. Using $POR(S)$, we can compute the Constraint Set (CS) and Non-Constraint Set (NCS) for S by the following formula.

1. $CS(S) = \{< a_i, a_j > \mid < a_i, a_j > \in POR(S), \text{ and } < a_j, a_i > \in POR(S)\}$;
2. $NCS(S) = \{< a_i, a_j > \mid < a_i, a_j > \in POR(S), \text{ but } < a_j, a_i > \notin POR(S)\}$.

Clearly, for a set of given sample S, $CS(S) \cap NCS(S) = \emptyset$. If $CS(S_1) \neq CS(S_2)$ (or $NCS(S_1) \neq NCS(S_2)$), then $S_1 \neq S_2$.

Definition 2. ***PS(P,s).*** *$PS(P, s)$ is a function in which P is a finite set of symbols and s is a string. Each symbol s_i of s in $PS(P, s)$ is defined as follows: $\pi_s(P, s_i) = s_i$ if $s_i \in P$; otherwise $\pi_s(P, s_i) = \varepsilon$. The return value of $PS(P, s)$ is a new string s' with ε removed.*

For example, let $P = \{b, c, r\}$ and $s = ebbdfc$. $s' = PS(P, s) = bbc$.

Definition 3. ***extended String (eS).*** *Let Σ be a finite set of terminal symbols. An eS is a finite sequence $s_1^{c_1} s_2^{c_2} \cdots s_n^{c_n}$, where $s_i \in \Sigma$ and $c_i \in \{1, ?, +, *\}$.*

Definition 4. ***Various CHAin Regular Expression with Interleaving (VCHARE).*** *Let Σ be a finite alphabet. A $VCHARE$ is a regular expression with interleaving over Σ in which each symbol occur once at most. It consists of a finite sequence of factors of two forms. One form is of $a_1^{c_1} \& a_2^{c_2} \& \cdots \& a_n^{c_n}$ where $n \geq 2$, $a_i \in \Sigma$ and $c_i \in \{1, ?, +, *\}$. The other form is of $f_1 f_2 \cdots f_m$ where $m \geq 1$. Each factor f_i is of the form of $(b_1|b_2|\cdots|b_n)$, $(b_1|b_2|\cdots|b_n)^?$, $(b_1|b_2|\cdots|b_n)^+$ or $(b_1|b_2|\cdots|b_n)^*$ where b_i has two forms: 1. terminal symbol a or a^+ with $|b_i| = 1$ for the first two forms; 2. for the last two forms, it can be an eS $s = a_1^{c_1} a_2^{c_2} \cdots a_n^{c_n}$ where $a_i \in \Sigma$ and $c_i \in \{?, *\}$ with $n \geq 1$.*

Clearly, $E_1 = a^? \& b^* \& c^+$ and $E_2 = a^?(b + c^+)(c^? d^* + e^?)^+$ are both *VCHAREs*.

3 Inference Algorithm

In this section, we will introduce the inference algorithm *GenVCHARE* for *VCHARE*. The algorithm is based on SOA and MIS.

We use the method *2T-INF* [15] to construct a SOA for S. It was proved that $L(SOA(S))$ is inclusion-minimal of S. Finding a maximum independent set from a graph G is a well-known NP-hard problem. Therefore we use the approximation method *clique_removal*() [10] to find the approximative results. *all_mis* is the set contained all maximum independent sets iteratively obtained from G using *clique_removal*(). *symbol*(A) is the set of all symbols occur in A. The main procedure of *GenVCHARE* is described as follows.

- Construct a graph $G(V, E) = SOA(S)$ using method *2T-INF* [15].
- For each node v with a self-loop, label it with v^+ and remove the self-loop. Update the graph G.

- If G is a strongly connected component, then return the result $v_1^{c_1}\&v_2^{c_2}\&\cdots\&v_n^{c_n}$ where $v_i \in V$ and assign the repetition operator $c_i \in \{1, ?, +, *\}$ using CRX [6]. Otherwise, continue to run the following steps.
- For each non-trival strongly connected component c_i, replace it with the return value of $RepairRE()$ as one new node. All relations with any node in c_i rebuild the relations with the new node.
- Assign level numbers for the new graph and compute all skip levels.
- Nodes of each level are turned into one or more chain factors. If there are more than one non-letter nodes (label with more than one terminal symbols) with the same ln, or if ln is a skip level, then ? is appended to every chain factor on that level.

Pseudo code for GenVCHARE ALT(C) can be found on the web site: http://lcs.ios.ac.cn/~zhangxl/.

Algorithm Analysis. For graph $G(V, E) = SOA(S)$, let $n = |V|$ and $m = |E|$. It costs time $O(n)$ to find all nodes with self-loops and $O(m + n)$ to find all NTSCCs. The time complexity of $clique\_removal()$ is $O(n^2 + m)$. For each NTSCC, computation of all_mis costs time $O(n^3 + m)$ and the topological sort for each mis costs time $O(m + n)$. The number of NTSCCs in a SOA is finite. Therefore computing all_mis for all NTSCCs also costs time $O(n^3 + m)$. Assigning level numbers and computing all skip levels will be finished in time $O(m+n)$. All nodes will be converted into specific chain factors of $VCHARE$ in $O(n)$. Therefore, the time complexity of $GenVCHARE$ is $O(n^3 + m)$.

Theorem 1. *Suppose that $\alpha = GenVCHARE(SOA(S))$ where S is a set of given sample. If there exists another VCHARE β such that $S \subseteq L(\beta) \subset L(\alpha)$, then $L(\beta) = L(\alpha)$.*

All detail proofs are omitted due to limited space.

4 Experiments and Analysis

In this section, we first investigate the proportion of $VCHARE$ based on real-world data, and then analyze our inference algorithm on $DBLP$ downloaded from the Web[1]. $DBLP$ is a Computer Science Bibliography corpus, a data-centric database of information on major computer science journals and proceedings. All our experiments were conducted on a machine with Intel Core i5-5200U@2.20 GHz, 4G memory, OS: Ubuntu 16.04. All codes were written in python 3.

[1] http://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html.

4.1 Usage of VCHARE in Practice

To investigate the proportion of *VCHARE* in practice, we crawled 29414 DTDs, 38554 XSDs and 4526 Relax NGs files from the Web and extracted 118242, 476804 and 509267 regular expressions from them respectively. The coverage proportions of subclasses: *VCHARE, SORE, DME, ME, Ghelli* [17], *SIRE* are shown in Fig. 1. Clearly, we can find out that the proportions of *VCHARE* are the highest for XSDs and Relax NG which are 94.95% and 95.28% respectively. For DTDs, the proportion (93.54% for *VCHARE*) is a little lower than *SORE* (96.69%). This is because interleaving operator is not supported in DTD. Interleaving is defined in an unlimited manner in Relax NG with any symbol in strings to interleave in any order while it is limited in XSD with only single symbols to interleave in any order. For example, $(ab^*)\&(c^+d^?)$ is not valid in XSD but it is allowed in Relax NG. Although interleaving defined in *SIRE* conforms to Relax NG, the proportion of *VCHARE* is still higher than *SIRE*. This means that in actual data, interleaving is used mostly in a quite simple and concise form. Therefore, *VCHARE* is more practical in real-world applications.

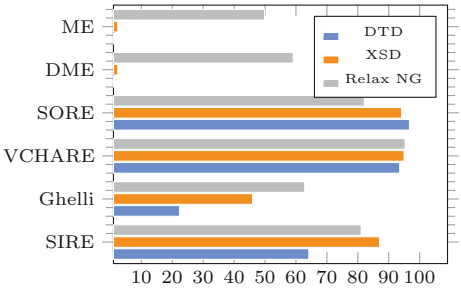

Fig. 1. Proportions of subclasses

4.2 Analysis of Inference Results Compared with *GenVCHARE*

In this section, we analyze the inference results by *GenECHARE* [12] (algorithm for inferring *eSimplified CHARE*), *Soa2Chare* [13], *Original Schema*, *Trang*,

Table 1. Results of inference using different methods on **inproceedings**

Sample size	From	Element name	ND	$\|RE\|$
1610138	DBLP	Inproceedings		
Methods	Regular expression			
1. Original Schema	$(a_1\|a_2\|a_3\|a_4\|a_5\|a_6\|a_7\|a_8\|a_9\|a_{10}\|a_{11}\|a_{12}\|a_{13}\|a_{14}\|a_{15}\|a_{16}\|a_{17}\|a_{18}\|a_{19}\|a_{20}\|a_{21}\|a_{22}\|a_{23})^*$		1	48
2. IntelliJ IDEA	$a_2^*(a_1\|a_3\|a_4\|a_5\|a_6\|a_{10}\|a_{11}\|a_{12}\|a_{13}\|a_{14}\|a_{15}\|a_{17}\|a_{18})^+$		1	31
3. Liquid Studio	$(a_1\|a_2\|a_3\|a_4\|a_5\|a_6\|a_{10}\|a_{11}\|a_{12}\|a_{13}\|a_{14}\|a_{15}\|a_{17}\|a_{18})^+$		1	30
4. Trang	$a_2^*(a_1\|a_3\|a_4\|a_5\|a_6\|a_{10}\|a_{11}\|a_{12}\|a_{13}\|a_{14}\|a_{15}\|a_{17}\|a_{18})^+$		1	31
5. Soa2Chare	$a_2^*(a_1\|a_3\|a_4\|a_5\|a_6\|a_{10}\|a_{11}\|a_{12}\|a_{13}\|a_{14}\|a_{15}\|a_{17}\|a_{18})^+$		1	31
6. GenEchare	$a_2^*(a_1^+\|a_3\|a_4^+\|a_5\|a_6\|a_{10}\|a_{11}\|a_{12}\|a_{13}\|a_{14}^+\|a_{15}^+\|a_{17}\|a_{18}^+)^+$		2	36
7. conMiner	$a_1^*a_{17}^?a_{14}^*\&a_2^*a_{11}^?a_4a_{10}^?\&a_3a_6\&a_5^?\&a_{12}\&a_{13}^*\&a_{18}^*a_{15}^*$		1	37
8. GenVCHARE	$a_2^*(a_1^*a_{17}^?a_{14}^*\|a_3a_{12}^?a_{15}^*\|a_4\|a_5\|a_6\|a_{13}^*\|a_{18}^*a_{11}^?a_{10}^?)^+$		2	40

conMiner [29] (algorithm for inferring SIRE), *IntelliJ IDEA* and *Liquid Studio* compared with *GenVCHARE* on *inproceedings, incollection, phdthesis, masterthesis*. Using two indicators: *Nesting Depth* [21] and *length of regular expressions* (the number of symbols together with operators), we only give the analysis of inferred regular expressions on *inproceedings* due to limited space reason. Analysis on other elements can be found on the web site: http://lcs.ios.ac.cn/~zhangxl/.

From Table 1, we can find that a_2 must occur in the first position if it appears. However, its position is not fixed in regular expressions inferred from methods 1, 3, 7 which lead to over-generalization.

5 Conclusion and Future Work

After a detailed analysis of real-world data, we propose a new subclass *VCHARE* of restricted regular expressions considering interleaving operator. Each terminal symbol in a *VCHARE* can only occur at most once. Compared with existing subclasses, *VCHARE* can cover more real-world data. This is useful for applications such as data process and integration and so on. Further, we proposed an inference algorithm *GenVCHARE* for *VCHARE* based on *SOA* and *MIS*. It is proved that regular expressions inferred by *GenVCHARE* are descriptive generalized. Experimental results show that regular expressions inferred by *GenVCHARE* is more accurate.

One future work is to consider constructing an automaton for regular expression with interleaving which is useful for schema inference. In addition, we will also study *SORE* extended with interleaving.

References

1. Abiteboul, S., Bourhis, P., Vianu, V.: Highly expressive query languages for unordered data trees. Theor. Comput. Syst. **57**(4), 927–966 (2015)
2. Benedikt, M., Fan, W., Geerts, F.: XPath satisfiability in the presence of DTDs. J. ACM **55**(2), 1–79 (2008)
3. Bernstein, P.A.: Applying model management to classical meta data problems. In: CIDR. vol. 2003, pp. 209–220. Citeseer (2003)
4. Bex, G.J., Gelade, W., Neven, F., Vansummeren, S.: Learning deterministic regular expressions for the inference of schemas from XML data. ACM Trans. Web **4**(4), 1–32 (2010)
5. Bex, G.J., Neven, F., Bussche, J.V.D.: DTDs versus XML schema: a Practical Study. In: International Workshop on the Web and Databases, pp. 79–84 (2004)
6. Bex, G.J., Neven, F., Schwentick, T., Tuyls, K.: Inference of concise DTDs from XML data. In: International Conference on Very Large Data Bases, Seoul, Korea, pp. 115–126, September 2006
7. Bex, G.J., Neven, F., Schwentick, T., Vansummeren, S.: Inference of concise regular expressions and DTDs. ACM Trans. Database Syst. **35**(2), 1–47 (2010)
8. Bex, G.J., Neven, F., Vansummeren, S.: Inferring XML schema definitions from XML data. In: International Conference on Very Large Data Bases, University of Vienna, Austria, pp. 998–1009, September 2007

9. Boneva, I., Ciucanu, R., Staworko, S.: Simple schemas for unordered XML. In: International Workshop on the Web and Databases (2015)
10. Boppana, R., Halldrsson, M.M.: Approximating maximum independent set by excluding subgraphs. Bit Numer. Math. **32**(2), 180–196 (1992)
11. Che, D., Aberer, K., Özsu, M.T.: Query optimization in XML structured-document databases. VLDB J. **15**(3), 263–289 (2006)
12. Feng, X.Q., Zheng, L.X., Chen, H.M.: Inference algorithm for a restricted class of regular expressions. Comput. Sci. **41**, 178–183 (2014)
13. Freydenberger, D.D., Kötzing, T.: Fast learning of restricted regular expressions and DTDs. Theor. Comput. Syst. **57**(4), 1114–1158 (2015)
14. Freydenberger, D.D., Reidenbach, D.: Inferring Descriptive Generalisations of Formal Languages. Academic Press Inc., Cambridge (2013)
15. Garcia, P., Vidal, E.: Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. IEEE Trans. Pattern Anal. Mach. Intell. **12**(9), 920–925 (2002)
16. Garofalakis, M., Gionis, A., Shim, K., Shim, K., Shim, K.: XTRACT: learning document type descriptors from XML document collections. Data Min. Knowl. Disc. **7**(1), 23–56 (2003)
17. Ghelli, G., Colazzo, D., Sartiani, C.: Efficient inclusion for a class of XML types with interleaving and counting. Inf. Syst. **34**(7), 643–656 (2009)
18. Gold, E.M.: Language identification in the limit. Inf. Control **10**(5), 447–474 (1967)
19. Grijzenhout, S., Marx, M.: The quality of the XML web. Web Semant. Sci. Serv. Agents World Wide Web **19**, 59–68 (2013)
20. Koch, C., Scherzinger, S., Schweikardt, N., Stegmaier, B.: Schema-based scheduling of event processors and buffer minimization for queries on structured data streams. In: Thirtieth International Conference on Very Large Data Bases, pp. 228–239 (2004)
21. Li, Y., Zhang, X., Peng, F., Chen, H.: Practical study of subclasses of regular expressions in DTD and XML schema. In: Li, F., Shim, K., Zheng, K., Liu, G. (eds.) APWeb 2016. LNCS, vol. 9932, pp. 368–382. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45817-5_29
22. Manolescu, I., Florescu, D., Kossmann, D.: Answering XML queries on heterogeneous data sources. In: International Conference on Very Large Data Bases, pp. 241–250 (2001)
23. Martens, W., Neven, F.: Typechecking top-down uniform unranked tree transducers. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, pp. 64–78. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36285-1_5
24. Martens, W., Neven, F.: Frontiers of tractability for typechecking simple XML transformations. In: ACM Sigmod-Sigact-Sigart Symposium on Principles of Database Systems, pp. 23–34 (2004)
25. Martens, W., Neven, F., Schwentick, T.: Complexity of decision problems for XML schemas and chain regular expressions. SIAM J. Comput. **39**(4), 1486–1530 (2013)
26. Melnik, S.: Generic Model Management: Concepts and Algorithms. Springer, Heidelberg (2004). https://doi.org/10.1007/b97859
27. Min, J.K., Ahn, J.Y., Chung, C.W.: Efficient extraction of schemas for XML documents. Inf. Process. Lett. **85**(1), 7–12 (2003)
28. Papakonstantinou, Y., Vianu, V.: DTD inference for views of XML data. In: Nineteenth ACM Sigmod-Sigact-Sigart Symposium on Principles of Database Systems, pp. 35–46 (2000)

29. Peng, F., Chen, H.: Discovering restricted regular expressions with interleaving. In: Cheng, R., Cui, B., Zhang, Z., Cai, R., Xu, J. (eds.) APWeb 2015. LNCS, vol. 9313, pp. 104–115. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25255-1_9
30. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**, 334–350 (2001)
31. Thompson, H.S.: XML schema part 1: structures. Recommendation **6**, 291–313 (2001)