Inference of a Concise Regular Expression Considering Interleaving from XML Documents

Xiaolan Zhang[1,2], Yeting Li[1,2], 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

Abstract. XML schemas are useful in various applications. However, many XML documents in practice are not accompanied by a schema or by a valid schema. Therefore, it is essential to design efficient algorithms for schema learning. Each element in XML schema has its content model defined by a regular expression. Schema learning can be reduced to the inference of restricted regular expressions. In this paper, we focus on learning restricted regular expressions with interleaving from a set of XML documents. The new subclass is named as *CHAin Regular Expression with Interleaving (ICHARE)*. Then based on single occurrence automaton (SOA) and maximum independent set (MIS), we introduce an inference algorithm *GenICHARE*. The algorithm is proved to infer a descriptive *ICHARE* from a set of given sample. At last, based on the data set crawled from the Web, we compare the coverage proportion of *ICHARE* compared with other existing subclasses. Besides, we analyze the conciseness of regular expressions inferred by *GenICHARE* based on *DBLP*. Experimental results show that *ICHARE* is more concise and useful in practice, and the inference algorithm is promising and effective.

Keywords: XML · Regular expression · Interleaving
Language learning · Algorithms

1 Introduction

As a main data exchange format, eXtensible Markup Language (XML) has been widely used in various applications on the Web [1]. XML schemas are convenient for data processing, automatic data integration, static analysis of transformations and so on [16,18,19]. REgular LAnguage for XML Next Generation (Relax

H. Chen—Work supported by the National Natural Science Foundation of China under Grant Nos. 61472405.

D. Phung et al. (Eds.): PAKDD 2018, LNAI 10938, pp. 389–401, 2018.
https://doi.org/10.1007/978-3-319-93037-4_31

NG) is a simple schema language recommended by OASIS[1]. However, many XML documents are not accompanied by a (valid) schema in practice. A survey [15] showed that XML documents on the Web with corresponding schema definitions only accounted for 24.8% in 2013, with the proportion of 8.9% for valid ones. 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 [4,6,12].

Gold [14] proposed a classical language learning model (*learning in the limit or explanatory learning*) and proved that the class of regular expressions can not be identifiable from finite positive data only, even for the class of deterministic regular expressions (proved by Bex et al. [2]). So researchers have turned to the study of restricted subclasses of regular expressions. Different from the emphasis of an exact representation of the target language in Gold-style learning, Freydenberger proposed another learning model called *descriptive generalization* in [10]. 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 the sample S if $\mathcal{L}(\alpha)$ is an inclusion-minimal generalization of S. It means that there is no $\beta \in \mathcal{D}$ such that $S \subseteq \mathcal{L}(\beta) \subset \mathcal{L}(\alpha)$.

Popular subclasses such as SORE [4], Simplified CHARE [4] (originally was called CHARE), eSimplified CHARE [9], CHARE [3] (originally was called simple regular expressions), eCHARE [20] were discussed in detail in [17]. They are all based on standard regular expressions. In data-centric applications, there may be no order constraint among siblings [1]. But the relative order within siblings may be still important. In such cases the interleaving is necessary. However, there is only a few work which support interleaving. In [7], Ciucanu and Staworko studied the unordered XML documents and proposed two unordered schema formalisms: *disjunctive multiplicity expressions* (DME) and *disjunction-free multiplicity expressions* (ME). However, they both do not support the concatenation operator. Peng [21] proposed a subclass *SIRE*, which supports the concatenation but not the union operator. Ghelli et al. [13] introduced a grammar considering interleaving: $T ::= \varepsilon | a^{[m,n]} | T + T | T \cdot T | T \& T$ where $m \in N \backslash \{0\}$ and $n \in N \backslash \{0\} \cup \{*\}$. This grammar requires each terminal symbol appear at most once and counter can only occur as a constraint for terminal symbols.

As for inference algorithms for subclasses, Bex et al. [4,5] gave two algorithms *RWR* and *CRX* for SORE and Simplified CHARE. However, regular expressions inferred by these two algorithms are not descriptive generalized. Freydenberger et al. [10] proposed algorithms $Soa2Chare$ and $Soa2Sore$ for Simplified CHARE and SORE respectively which satisfy *descriptive generalization*. Ciucanu and Staworko introduced an algorithm for *DME* based on *max clique* [7]. Peng et al. [21] developed an approximate and heuristic solution to infer an approximate descriptive generalized $SIRE$.

In present paper, we propose a new subclass of restricted regular expressions called *CHAin regular expression with interleaving (ICHARE)* and give an inference algorithm *GenICHARE* to infer a descriptive generalized *ICHARE* based

[1] https://www.oasis-open.org/standards.

on *single occurrence automaton* (SOA) and *maximum independent set* (MIS). First, an SOA is constructed for given sample S. Then each non-trival strongly connected component is renamed by the return value of *Repair()*. Next, assign level numbers for the new graph. Finally, all nodes of each level will be converted to one or more chain factors.

The main contributions of this paper are as follows.

1. We propose a subclass of restricted regular expressions: CHAin regular expression with interleaving (*ICHARE*).
2. We design an inference algorithm *GenICHARE* to infer descriptive generalized *ICHAREs*.
3. A series of experiments were conducted. We compare the coverage proportion of *ICHARE* with existing subclasses. Based on the data set *DBLP*, we analyze the conciseness of regular expressions inferred by *GenICHARE* compared with other methods. The experimental results show that *ICHARE* is more useful and concise in real-world applications.

The paper is organized as follows. Section 2 is the definitions. The inference algorithm *GenICHARE* is introduced in Sect. 3. Experiments are given in Sect. 4. Section 5 is the conclusions.

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 is inductively defined as follows: ε or $a \in \Sigma$ is a regular expression. 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. $E^?$ and E^+ are used as abbreviations of $E|\varepsilon$ and EE^*, respectively. $L(E)$ is the language generated by the regular expression E. We use $sym(E)$ to denote the set of terminal symbols occurred in E and $term(v)$ to denote all symbols of the form a or a^+ where $a \in \Sigma$.*

Let $u = au'$ and $v = bv'$ where $a, b \in \Sigma$ and $u', v' \in \Sigma^*$, then $u\&v = a \cdot (u'\&v) \cup b \cdot (u\&v')$. For example, strings accepted by $(abc)\&(d)$ is the set $\{abcd, dabc, adbc, abdc\}$. Regular expressions with interleaving, in which each terminal symbol occurs at most once is called *ISORE* extended from *SORE* in [4]

Definition 2 ***Single Occurrence Automaton (SOA)*** *[5]. Let Σ be a finite alphabet. src,snk do not occur in Σ. A single occurrence automaton over Σ is a finite directed graph $G = (V, E)$ which satisfies the following two conditions.*

1. *$src, snk \in V$, and $V \subseteq \Sigma \cup \{src, snk\}$. All nodes in G are distinct.*
2. *src is the starting node which only has the outgoing edges while the accepting node snk only has the incoming edges. Each node in the set of $\{V \setminus \{src, snk\}\}$ lies on a path from src to snk.*

A *generalized* SOA over Σ is a finite directed graph in which each node $v \in V \setminus \{src, snk\}$ is an ISORE and all nodes are pairwise alphabet-disjoint ISOREs.

level number (ln) and skip level (sl) [10]. For a given directed acyclic graph $G = (V, E)$ of a SOA, the level number of node v is the longest path from src denoted by $ln(v)$. $ln(src)$ is 0. If there exists a path $v_1 \rightarrow_G v_2$ such that $ln(v_1) < ln(v)$ and $ln(v) < ln(v_2)$, then $ln(v)$ is a skip level.

Independent Set (IS). An independent set is a set S of nodes in a graph G, in which there is no edge between every two nodes in S. A *maximum independent set (MIS)* is an independent set with largest possible size for a given graph G where the size means the number of nodes in an *IS*.

Let $POR(S)$ denote the set of all partial order relations of each string in S. Then the Constraint Set (CS) and Non-Constraint Set (NCS) for a set of given sample S are computed as follows.

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

Clearly, for S, its Constraint Set and Non-constraint Set are both unique. If $CS(S_1) \neq CS(S_2)$ (or $NCS(S_1) \neq NCS(S_2)$), then $S_1 \neq S_2$.

Definition 3 ***PSubstring (P, s).*** *PSubstring (P, s) is a function in which P is a finite set of symbols and s is a string. It returns a new string each symbol of which is computed as follows:* $\pi_s(P, s_i) = s_i$ *if* $s_i \in P$; $\pi_s(P, s_i) = \varepsilon$ *otherwise.*

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

Definition 4 ***CHAin Regular Expression with Interleaving (ICHARE).*** *Let Σ be a finite alphabet. An ICHARE over Σ is a ISORE. It is of the form $(f_1 f_2 \cdots f_n)$ where $n \geq 1$. Each factor f_i has two possible forms. One is $(b_1|b_2|\cdots|b_m)$, $(b_1|b_2|\cdots|b_m)^?$, $(b_1|b_2|\cdots|b_m)^*$ or $(b_1|b_2|\cdots|b_m)^+$ where $m \geq 1$, b_i can be a or a^+ $(a \in \Sigma)$. The other is $s_1^{c_1} \& s_2^{c_2} \& \cdots \& s_t^{c_t}$ where $t \geq 2$, $s_i \in \Sigma^*$ and $c_i \in \{1, ?, *, +\}$.*

For example, $E = (a_1|a_2^+)((a_3^? a_4)\&(a_5 a_6))$ is an *ICHARE*.

3 Inference Algorithm GenICHARE

For a set of given sample S, *GenICHARE* can infer a descriptive generalized *ICHARE* α such that $S \subseteq L(\alpha)$. $sym(A)$ is the set of terminal symbols occur in A. *all_mis* means the set of all maximum independent sets iteratively obtained from a graph G. The main procedures are illustrated as follows.

1. Construct the graph $G\ (V, E) = SOA(S)$ for S.
2. For each node v with a self-loop, remove the self-loop and rename it with v^+. For each non-trival strongly connected component (NTSCC), replace it with the return value of *Repair()*. Relations with any node in NTSCC rebuild the relation with the new node.

3. Compute the level numbers for the new graph and find all skip levels.
4. Nodes of each level number ln are turned into one or more chain factors. If there are more than one non-letter node (node with more than one terminal symbols), or if ln is a skip level, then ? is appended to every chain factor on ln.

Now is the pseudo-code inference algorithm *GenICHARE*. *Repair*() is the subfunction which is to transform each NTSCC into a regular expression with interleaving. $setln(G)$ is to assign level number to each node. $issl(ln)$ returns *true* if ln is a skip level and returns *false* otherwise. For a given SOA $G = (V, E)$, the number of NTSCCs, the set of all_mis and level numbers are all finite. Then *Repair*() will stop after finite steps. Therefore, Algorithm 1 will also stop in finite steps.

Algorithm 1. GenICHARE(S)

Input: A set of given sample S
Output: An *ICHARE* R

Construct $G(V, E) = SOA(S)$ using method *2T-INF* [11];
For each node v with self-loop, remove the self-loop and rename it with v^+; for each NTSCC, replace it with $Repair(NTSCC, S)$. Relations with any node in NTSCC, rebuild the relation with the new node; Update the graph G;
$G.setln()$,$R \leftarrow \varepsilon$,$level = 1$;
while $level \leq (ln(G.sink)) - 1$ **do**
  $V_T \leftarrow$ all vertices with $level$ and $length(sym(v)) \geq 2$;
  $V_S \leftarrow$ all vertices with $level$ and $length(sym(v)) = 1$;
  for each $v_T \in V_T$ **do**
    if $G.issl(level)$ or $(|V_T| + |V_S|) > 1$ **then**
      $R \leftarrow R \cdot v_T^?$
    else
      $R \leftarrow R \cdot v_T$
    end if
  end for
  if $|V_S| > 0$ **then**
    if $G.issl(level)$ or $|V_T| > 0$ **then**
      $R \leftarrow R \cdot AIT(V_S)^?$
    else
      $R \leftarrow R \cdot AIT(V_S)$
    end if
  end if
  $level = level + 1$
end while
return R

Algorithm 2. Repair(V,S)

Input: A node set V and a set of given sample S
Output: A repaired regular expression $newRE$

Construct the new sample $S' \leftarrow \bigcup_{s \in Strings} PSubstring(V, s)$;
if $CS(S') == \emptyset$ **then**
  return RE
else
  $G = Graph(CS(S'))$;
  while $G.nodes()! = \emptyset$ **do**
    $v = clique\_removal(G)$; $all\_mis.append(v)$;
  end while
  for each $M_i \in all\_mis$ **do**
    $T.append(topological_sort(M_i))$;
  end for
  for nodes in $NCS(S')$ but not in $CS(S')$, add them in M_i with largest size. Determine the orders of symbols in $M_i \in T$ under condition of $NCS(S')$;
  add repetition operator $\{1, *, ?, +\}$ for each symbol in T using algorithm CRX[4]
  $newRE \leftarrow$ Combine all elements in T using &;
  return $newRE$
end if

Time Complexity. Let $G(V, E) = SOA(S)$, $n = |V|$ and $m = |E|$. It costs $O(n)$ to find all nodes with loops and $O(m+n)$ to find all NTSCCs using depth-first search algorithm [8]. The time complexity of $clique\_removal()$ is $O(m+n^2)$. $setln()$ can be finished in time of $O(m + n)$ using breadth-first search [8]. For each NTSCC, computation of all_mis costs time $O(m + n^3)$ and the topological sort for each subgraph M_i costs time $O(m+n)$. The number of NTSCCs is finite. Then computing all_mis for all NTSCCs also cost time $O(m + n^3)$. All nodes will be converted into specific chain factors of $ICHARE$ in $O(n)$. Therefore, the time complexity of $GenICHARE$ is $O(m + n^3)$.

Theorem 1. *For a set of given string sample S, $\alpha = GenICHARE(SOA(S))$. If $S \subseteq L(\beta) \subset L(\alpha)$, then $L(\beta) = L(\alpha)$ holds for every ICHARE β.*

Proof. We construct SOA for S, α, β as G_s, G_α, G_β respectively. Clearly, we have $sym(G_s) = sym(G_\alpha) = sym(G_\beta)$. Let $\alpha = \alpha_1\alpha_2 \cdots \alpha_n$, $\beta = \beta_1\beta_2 \cdots \beta_m$. Now we first consider α_1. α_1 contains all nodes with $level = 1$. We use V_S and V_T to denote the sets with only one terminal symbol and multiple terminal symbols, respectively. 6 cases have to be considered. 1. $V_S \neq \emptyset$, $V_T = \emptyset$, 1 is not a skip number. 2. $V_S \neq \emptyset$, $V_T = \emptyset$, 1 is a skip level. 3. $V_S = \emptyset$, $|V_T| = 1$, 1 is not a skip level. 4. $V_S = \emptyset$, $|V_T| = 1$, 1 is a skip number. 5. $V_S = \emptyset$, $|V_T| > 1$. 6. $V_S \neq \emptyset$ and $V_T \neq \emptyset$.

1. $V_S \neq \emptyset$, $V_T = \emptyset$, 1 is not a skip number.

Let $V_S = \{v_1, v_2, \cdots, v_k\}$. According to Algorithm 1, we know that $\alpha_1 = (v_1|v_2|\cdots|v_k)$ and there is no edge between any two nodes. $sym(\alpha_1) = sym(\beta_1)$, otherwise edges from src of G_α and G_β are different which will cause a contradiction with $S \subseteq L(\beta) \subseteq L(\alpha)$. & and $\cdot$ operators must not appear in β_1.

Because these two operators would lead to an edge $e_{<v_i,v_j>}$ that is not in α_1. Therefore, β_1 only have union operator. Because $ln = 1$ is not a skip level, there does not exist an edge from src to a node in $\Sigma \setminus \{src\} \setminus symbol(\alpha_1)$. Now we have $\beta_1 = \alpha_1$ and therefore $L(\beta_1) = L(\alpha_1)$.

2. $V_S = \emptyset$, $V_T \neq \emptyset$, 1 is a skip level.

According to the proof of case 1, we can conclude that $term(\alpha_1) = term(\beta_1)$. β_1 only have union operator. At the same time, there exists an edge from src to some node $v \in \Sigma \setminus \{src\} \setminus symbol(\alpha_1)$. $\alpha = GenICHARE(G_s)$, $src \rightarrow_S v$ holds. Because G_s is a subgraph of G_{β_1}, $src \rightarrow_{\beta_1} v$ holds also. Therefore, $\beta_1 = \alpha_1$. $L(\beta_1) = L(\alpha_1)$.

3. $V_S = \emptyset$, $|V_T| = 1$, 1 is not a skip level.

Let $V_T = \{V_T^1\}$, $V_T^1 = (v_1 \& v_2 \& \cdots \& v_k)^+$. V_T is a NTSCC of G_α. According to the theorem, it is also a NTSCC of G_β and G_s. In this situation, we use Algorithm 2 to decide whether it needs to be repaired.

- α_1 can not be repaired. Then $\alpha_1 = (v_1|v_2|\cdots|v_k)^+$. The CS of $S' = \bigcup_{s \in S} PSubstring(sym(\alpha_1), s)$ is an empty set. If β_1 has & operator, it must generate two edges $e_{<v_i,v_j>}$ and $e_{<v_j,v_i>}$ at the same time which will lead to a contradiction of $L(G_\beta) \subseteq L(G_\alpha)$. If β_1 has concatenation operator, then the order of two nodes v_i, v_j is fixed. That means in G_β, $v_i \rightarrow^+ v_j$ and $v_j \rightarrow^+ v_i$ can not hold at the same time. Therefore it leads to a contradiction that $G_s \subseteq G_\beta$ because V_T is a NTSCC of G_s.
- α_1 can be repaired. In this situation, β_1 does not has union operator. Otherwise, there is a string $s \in L(G_\beta)$ with a substring aaa but $s \notin L(G_\alpha)$.
  - α_1 only has & operator. According to $GenICHARE$, the NCS of α_1 is empty. The CS of α_1 contains all nodes of $sym(\alpha_1)$. If β_1 has concatenation operator, the CS of β_1 is different from that of α_1. This will lead to the conclusion that they are generated from different string samples, which is not right. Therefore, β_1 only has & operator. According to the theorem, the languages of $L(\alpha_1)$ is the minimal-inclusion. Because $L(G_s) \subseteq L(G_\beta) \subset L(G_\alpha)$ and each terminal symbol can only occur once at most, $L(G_{\beta_1}) = L(G_{\alpha_1})$ must hold.
  - α_1 has concatenation and interleaving operators. If β_1 does not has concatenation operator, the CS of β_1 is larger than that of α_1 which leads to the conclusion that they are generated by different string samples, which is wrong. Therefore, β_1 has two operators $\cdot$ and &. And their CS and NCS must be the same because they are both generated from S. Let $\alpha_1 = ABC$ where A and C only have concatenation and all symbols in $sym(A)$ must occur before that of $sym(B)$ and $sym(C)$. Similarly, all symbols in $sym(C)$ must occur after that of $sym(A)$ and $sym(B)$. Due to $L(G_s) \subseteq L(G_\beta) \subset L(G_\alpha)$, we can conclude that there must exist A' and B' with $sym(A) = sym(A')$ and $sym(B) = sym(B')$ in which A' and B' only has concatenation operator. For B in α_1, according to the proof above, $B = B'$. Otherwise, their CSs must be different. Therefore, we can conclude that $\beta_1 = \alpha_1$ and $L(G_{\beta_1}) = L(G_{\alpha_1})$.

4. $V_S = \emptyset$, $|V_T| = 1$, 1 is a skip number.

Similar as the proof in case 3 and case 2, we can easily come to the conclusion that $L(G_{\beta_1}) = L(G_{\alpha_1})$.

5. $V_S = \emptyset$, $|V_T| > 1$.

In this case, there are more than one NTSCC in G_{α_1}. For each NTSCC $C^i_{\alpha_1}$, we can prove that there exists a $\beta_m = \alpha^i_1$ where α^i_1 is the expression of $C^i_{\alpha_1}$ using the same proof in case 3. Each α^i_1 must be added with ?. Otherwise, there exists one node $v \in V_T$ with no outgoing edge. Therefore each β_m must have choice operator ? in order to accept all strings in sample S. This leads to the conclusion that $L(G_{\beta_1}) = L(G_{\alpha_1})$.

6. $V_S \neq \emptyset$ and $V_T \neq \emptyset$.

Obviously $L(G_{\beta_1}) = L(G_{\alpha_1})$ similar with the proof of case 1 and 5.

From the analysis above, we can conclude that there exists a i where $1 \leq i \leq m$ that $L(\alpha_1) = L(\beta_1 \cdots \beta_i)$ holds. Let $\alpha' = \alpha_2 \cdots \alpha_n$ and $\beta' = \beta_{i+1} \cdots \beta_m$. For each string $s \in S$, if $symb(s) \cap symb(\alpha_1) = \emptyset$, then $s \in S'$. If $symb(s) \cap symb(\alpha_1) \neq \emptyset$, then replace all alphabet in $symb(\alpha_1)$ as ε and put the new string in set S'. We construct $G_{S'} = SOA(S')$. α_1 is absorbed by src and the new starting node becomes $src \cdot \alpha_1$. Using the same proof procedure above, we can conclude that there exist a k such that $\alpha_2 = \beta_{j+1} \cdots \beta_{j+k}$. Go on the same operation until the graph G has only two nodes $src \cdot \alpha_1 \cdots \alpha_n$ and snk, and $\alpha_n = \beta_t \cdots \beta_m$, then we have $L(\alpha) = L(\beta)$.

4 Experiments

4.1 Usage of *ICHARE* in Practice

We first investigated the usage and proportion of *ICHARE*(denoted as G in the Fig. 1) compared with SORE(A), Simplified CHARE(B), eSimplified CHARE(C), CHARE(D), eCHARE(E), SIRE(F) based on real-world data set, and then validated our inference algorithm *GenICHARE* on DBLP[2]. All our experiments were conducted on a machine with Intel Core i5-5200U@2.20GHz, 4G memory, OS: Ubuntu 16.04. All codes were written in python 2.7.

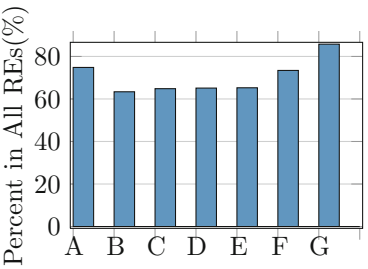

Fig. 1. Coverage proportions of subclasses

Our data set contains 13946 Relax NG files crawled from 268 Web sites, Maven[3], and 171 GitHub repositories with 509267 regular expressions extracted from these files. From Fig. 1, we can find that the coverage proportion of *ICHARE* is the highest (85.78%) among these subclasses.

[2] http://dblp.uni-trier.de/xml/.
[3] http://repo1.maven.org/maven2/.

4.2 Analysis of Inference Results Among Different Methods

DBLP is Data-centric database of information on major computer science journals and proceedings. We downloaded *dblp-2015-03-02.xml.gz*.

The results are shown in Tables 1 and 2 inferred by methods: Original Schema, Exact Minimal Schema, Trang[4], InstanceToSchema[5] and *GenICHARE*. The first column is the elements names with corresponding size of strings (considering both large and small data sets). The second column is regular expressions inferred by different methods. In both tables, we use "," to denote the concatenation operator. Method of Exact Minimal Schema uses the exact MIS to infer a regular expression. Trang can infer a schema from one or more XML documents. InstanceToSchema is a Relax NG schema generator from XML documents.

We analyze the inferred regular expressions from two aspects: conciseness and descriptive generalization. For conciseness, regular expressions inferred by the Original Schema, Trang and *GenICHARE* are in accordance with chain regular expressions. This form is concise and has a better readability while the rest two have no notable structure features. For descriptive generalization, we discuss the experimental inference results from the following four cases.

(1) Language generated by the inferred regular expression must contain all strings of given sample. Original Schema, Exact Minimal Schema, Trang and *GenICHARE* all satisfy this requirement while InstanceToSchema does not. Take *inproceedings* for example, the regular expression inferred by InstanceToSchema is $ee^*\&note^?\&editor^*\&year\&author^*\ \&title\&cdrom\&url^?\ \&number^?\&pages^?\&month^?\&cite^*\&booktitle\&crossref^*$. *cdrom* in this regular expression can occur only once. But we find out that the word *cdrom* appeared more than once in 1610138 strings i.e. $w = author \cdot title \cdot pages \cdot year \cdot booktitle \cdot url \cdot ee \cdot cdrom \cdot cdrom \cdot cite \cdot cite \cdot cite \cdot cite \cdot cite \cdot cite \cdot cite$.

(2) It is better for inferred regular expressions to support all binary operators ($\cdot$, |, &) which will have a higher practicality. However, regular expressions inferred by Exact Minimal Schema do not support union operator. Trang supports the generation of Relax NG schema using XML documents. But in fact it is almost equivalent to DTD. In other words, it does not support the regular expressions with interleaving. InstanceToSchema supports the interleaving operator but not concatenation and union. *GenICHARE* supports all operators which could generate real Relax NG schemas.

(3) Improper use of union will lead to the over-generalization for the given sample, which will generate many invalid XML documents. This shortage could be found in Original Schema and Trang. Take element of *article* for example. Regular expression inferred by Trang is $editor^*\cdot(author|booktitle|cdrom|cite|cros\ sref|ee|journal|month|note|number|pages|publisher|title|url|volume|year)^+$. But the title and published year for an actual article are both unique. This shortage is also

[4] http://www.thaiopensource.com/relaxng/trang.html.
[5] http://www.xmloperator.net/i2s/.

Table 1. Inference results using different methods on DBLP

name size	1.Original Schema; 2.Exact Minimal Schema 3.Result of Trang; 4.Result of InstanceToSchema; 5.Result of GenICHARE
phdt- hesis	(author|editor|title|booktile|pages|year|address|journal|volume|number|month |url|ee|cdrom|cite|publisher|note|crossref|isbn|series|school|chapter)*
6953	school?&author+,title,pages?,publisher?,series?,number?,month?,volume?, ee*,url?&year,isbn?,note*
6953	author+,title,(isbn|month|number|pages|publisher|school|series|volume|year) +,(ee|note|url)*
6953	ee*&note*&year&author+&isbn?&title&url?&number?&volume?&pages? &month?&school?&series?&publisher?
6953	author+,title,(year,month?,volume?,isbn?&school?&pages?,publisher?, series?,number?),(ee*,url&note?)
master- sthesis	(author|editor|title|booktile|pages|year|address|journal|volume|number|month |url|ee|cdrom|cite|publisher|note|crossref|isbn|series|school|chapter)*
9	ee?&author,title,year,school,url?
9	author,title,year,school,(ee|url)*
9	ee?&year&author&title&url?&school
9	author,title,year,school,(ee?&url?)
www	(author|editor|title|booktile|pages|year|address|journal|volume|number|month |url|ee|cdrom|cite|publisher|note|crossref|isbn|series|school|chapter)*
1557527	title?&url*,cite*&crossref?,author*,note*,editor*,booktitle?,ee?,year?
1557527	(crossref|editor*|author*),(booktiltle|cite|ee|note|title|url|year)*
1557527	ee?&editor*&note*&year?&author*&cite*&title?&booktitle?&crossref?&url*
1557527	(editor+|author+|crossref)?,(note*,year?&ee?,booktitle?,url*,cite*&title?)

reflected in the result of Original Schema but rest methods do not have this problem.

(4) Improper use of & without considering the actual orders in real-world data will also lead to over-generalization. Take the element of *masterthesis* for example. Regular expressions inferred by Exact Minimal Schema, Trang, InstanceToSchema and *GenICHARE* are $ee^?\&author{\cdot}title{\cdot}year{\cdot}school{\cdot}url^?$; $author \cdot title \cdot year \cdot school \cdot (ee|url)^*$; $ee^?\&year\&author\&title\&url^?\&school$ and $author \cdot title \cdot year \cdot school \cdot (ee^?\&url^?)$. Compared with the first result, it is clear that the form of third result will lead to more redundant XML documents. The results of Trang and *GenICHARE* reveal that orders of first four elements are fixed. Therefore, the result inferred by InstanceToSchema is not good enough. From the investigation of *mastersthesis*, the combinations of *ee* and *url* ($ee \cdot url$, $url \cdot ee$, *ee*, *url* or ε) can only occur in the last position. Therefore, regular expression inferred by *GenICHARE* is reasonable and descriptive generalized.

The proper use of interleaving & can ensure the actual orders of elements at the beginning. Therefore, the optimization is the significant feature of our algorithm *GenICHARE*.

Table 2. Inference results using different methods on DBLP

name size	1.Original Schema; 2.Exact Minimal Schema 3.Result of Trang; 4.Result of InstanceToSchema; 5.Result of *GenICHARE*
article	(author|editor|title|booktile|pages|year|address|journal|volume|number|month |url|ee|cdrom|cite|publisher|note|crossref|isbn|series|school|chapter)*
1270262	booktitle?,cdrom?,cite*,note?&number?&publisher?,month?,ee*&title ,journal?&editor*,volume?&pages?&author*,url?&year?,crossref?
1270262	editor*,(author|booktitle|cdrom|cite|crossref|ee|journal|month|note|number |pages|publisher|title|url|volume|year)+
1270262	ee*&note?&editor*&year?&author*&title&cdrom?&url?&volume?&number? &journal?&pages?&month?&cite*&publisher?&booktitle?&crossref?
1270262	editor*,(title,booktitle?,volume?,cite*&note?,cdrom?&year?&author*, publisher?,month?,url?&journal?&pages?&number?,crossref?&ee*)
inproce- edings	(author|editor|title|booktile|pages|year|address|journal|volume|number|month |url|ee|cdrom|cite|publisher|note|crossref|isbn|series|school|chapter)*
1610138	title,year&ee*&editor*,month?,booktitle,number?&url?&crossref*,cite* &pages?&author*,note?,cdrom*
1610138	editor*,(author|booktitle|cdrom|cite|crossref|ee|month|note|number|pages|title |url|year)+
1610138	ee*&note?&editor*&year&author*&title&cdrom&url?&number?&pages? &month?&cite*&booktitle&crossref*
1610138	editor*,(pages?&booktitle&ee*&crossref*,month?,number?&author*,note?, cdrom*&title,url?,cite*&year)
proce- edings	(author|editor|title|booktile|pages|year|address|journal|volume|number|month |url|ee|cdrom|cite|publisher|note|crossref|isbn|series|school|chapter)*
26502	isbn*&series*&title,note*&publisher?&editor*,pages?,crossref?,ee*,number?& booktitle?&year?&url*&author*,journal?,address?,volume?,cite*
26502	author*,(booktitle|crossref|editor|ee|isbn|journal|note|number|pages|publisher |series|title|url|volume|year|address)+,cite*
26502	ee*&note*&editor*&year?&author*&isbn*&title&url*&volume?&number?& journal?&pages?&cite*&series*&publisher?&booktitle?&crossref?&address?
26502	author*,(journal?&volume?&number?&pages?&note*&address?&title,crossref ?,ee*&publisher?&isbn*&series*&editor*&booktitle?&year?&url*),cite*
book	(author|editor|title|booktile|pages|year|address|journal|volume|number|month |url|ee|cdrom|cite|publisher|note|crossref|isbn|series|school|chapter)*
11600	series*&publisher?&booktitle?,url*,school?&ee*&pages?&isbn*,cdrom?,cite* ,note*&author*,editor*,title,volume?,month?&year
11600	(editor*|author*),title,(booktitle|cdrom|cite|ee|isbn|month|note|pages |publisher|school|series|url|volume|year)+
11600	ee*&editor*&note*&year&author*&isbn*&title&cdrom?&url*&volume? &pages?&month?&school?&series*&publisher?&cite*&booktitle?
11600	(editor+|author+)?,title,(series*&booktitle?,url*,school?&year&publisher? &isbn*&ee*&cdrom?,cite*,volume?,note*&pages?,month?)
incoll- ection	(author|editor|title|booktile|pages|year|address|journal|volume|number|month |url|ee|cdrom|cite|publisher|note|crossref|isbn|series|school|chapter)*
31260	ee?&author*,title,year,chapter?,number?,isbn?,cite*,note?,cdrom?&booktile ,publisher?,url?&pages?&crossref?
31260	author*,title,(booktitle|crossref|ee|isbn|number|pages|publisher|url|year| chapter)+,(cdrom|note|cite*)
31260	ee?&note?&year&author*&isbn?&title&cdrom?&url?&number?&pages?& cite*&publisher?&booktile&crossref?&chapter?
31260	author*,title,(pages?&ee?&booktitle,publisher?,number?,isbn?,chapter?, url?&crossref?&year),(cite+|note|cdrom)?

From the analysis above, the subclass *ICHARE* has a higher coverage proportion based on real-world data. This means that it will have a better practicality in actual applications. Regular expression inferred by *GenICHARE* are more concise and satify descriptive generalization.

5 Conclusion and Future Work

Based on real-world data, we proposed a new subclass *ICHARE* of restricted regular expressions with interleaving. Then we introduced an efficient algorithm *GenICHARE* to infer a descriptive generalized *ICHARE*. A series of experiments were conducted to compare the coverage of *ICHARE* with other existing subclasses based on real-world data. Besides, based on the data of *DBLP*, we analyse the conciseness of regular expressions inferred by *GenICHARE* and compare with other methods. Experimental results show that *ICHARE* is more useful in practice with higher coverage proportion than other subclasses. Regular expressions inferred by *GenICHARE* perform better from the aspects of conciseness and descriptive generalization than methods of Original Schema, Exact Minimal Schema, Trang and InstanceToSchema.

In the future, we will consider constructing the automata for regular expressions with interleaving and study the inference algorithms for subclass *k-ORE* (each terminal symbol can occur k times at most).

References

1. Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web: from Relations to Semistructured Data and XML. Morgan Kaufmann, San Francisco (2000)
2. 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)
3. 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)
4. 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, September, pp. 115–126 (2006)
5. 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)
6. 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, September, pp. 998–1009 (2007)
7. Boneva, I., Ciucanu, R., Staworko, S.: Simple schemas for unordered XML. In: International Workshop on the Web and Databases (2015)
8. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn., pp. 1297–1305 (2001)
9. Feng, X.Q., Zheng, L.X., Chen, H.M.: Inference Algorithm for a Restricted Class of Regular Expressions, vol. 41. Computer Science (2014)
10. Freydenberger, D.D., Kötzing, T.: Fast learning of restricted regular expressions and DTDs. Theory Comput. Syst. **57**(4), 1114–1158 (2015)

11. 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)
12. Garofalakis, M., Gionis, A., Shim, K.: XTRACT: learning document type descriptors from XML document collections. Data Min. Knowl. Discov. **7**(1), 23–56 (2003)
13. 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)
14. Gold, E.M.: Language Identification in the limit. Inf. Control **10**(5), 447–474 (1967)
15. Grijzenhout, S., Marx, M.: The quality of the XML web. Web Semant. Sci. Serv. Agents World Wide Web **19**, 59–68 (2013)
16. Koch, C., Scherzinger, S., Schweikardt, N., Stegmaier, B.: Schema-based scheduling of event processors and buffer minimization for queries on structured data streams. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, vol. 30, pp. 228–239. VLDB Endowment (2004)
17. 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
18. Manolescu, I., Florescu, D., Kossmann, D.: Answering XML queries on heterogeneous data sources. VLDB **1**, 241–250 (2001)
19. 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
20. Martens, W., Neven, F., Schwentick, T.: Complexity of decision problems for XML schemas and chain regular expressions. Siam J. Comput. **39**(4), 1486–1530 (2009)
21. 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