

# Learning restricted regular expressions with interleaving from XML data

## 1 Inference Algorithm

For the set of given sample  $S$ ,  $L(SOA(S))$  is a minimal-inclusion generalization of  $S$  using  $2T-INF$  [2]. Finding a *maximum independent set* (MIS) for a graph  $G$  is NP-hard proposed in [3]. Hence we use the method *clique\_removal()* [1] to find an approximate result. *all\_mis* is the set contained all MISs iteratively obtained from  $G$ . *sym(A)* is the set of all symbols occurring in  $A$ . *G.setln()* is to assign each node a level number *ln*. *G.isSL(i)* returns *true* if  $i$  is a skip level and *false* otherwise. *Combine(V, "|")* (or *Combine(V, "&")*) is to combine all elements in  $V$  with union ( $|$ ) (or interleaving ( $\&$ )) operator. The input is  $S$  and the output is an *ESIRE*  $R$ . The main procedures and the pseudo-code of the algorithm *GenESIRE* are as follows.

1. Construct a graph  $G(V, E)=SOA(S)$  for  $S$  using  $2T-INF$  [2].
2. For each node  $v$  with a self-loop, rename it with  $v^+$  and remove the self-loop.
3. For each non-trivial strongly connected component NTSCC, call function *Repair(NTSCC, S)*. Replace the NTSCC with a new node and label it with the return value of *Repair(NTSCC, S)*. All relations with any node in NTSCC of  $G$  rebuild the relations with the new node.
4. Assign the level numbers for the new graph and compute all the skip levels.
5. Nodes of each level are converted into one or more chain factors.

**Algorithm Analysis** For a graph  $G(V, E)=SOA(S)$ , let  $n=|V|$  and  $m=|E|$ . It costs time  $O(n)$  to find all nodes with 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  $O(n^3 + m)$ . For each *mis*, there is no NTSCCs at all. Hence *Repair()* only costs time  $O(n^3 + m)$ . The number of NTSCCs in a SOA is finite. Then computing *all\_mis* for all NTSCCs also costs time  $O(n^3 + m)$ . Assigning level numbers and computing the skip numbers will be finished in time  $O(m + n)$ . All nodes will be converted into specific chain factors of *ESIRE* in  $O(n)$ . Therefore, the time complexity of *GenESIRE* is  $O(n^3 + m)$ .

**Theorem 1.** For a given graph  $G(V, E)=SOA(S)$ , a *ESIRE*  $R$  will be inferred by the algorithm *GenESIRE* and  $L(G) \subseteq L(R)$ .

**Theorem 2.** Let  $\alpha=GenESIRE(SOA(S))$  where  $S$  is a set of given sample. If there exists another *ESIRE*  $\beta$  such that  $S \subseteq L(\beta) \subset L(\alpha)$ ,  $L(\beta)=L(\alpha)$  must hold.

*Proof.* We construct SOA for  $S$ ,  $\alpha$ ,  $\beta$  as  $G_s$ ,  $G_\alpha$ ,  $G_\beta$  respectively. Obviously, 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  $\alpha_1$ .  $\alpha_1$  contains all nodes with  $ln=1$ . We use  $V_S$  and  $V_T$  to denote the sets of nodes with each node containing only one terminal symbol and multiple terminal symbols, respectively.

---

**Algorithm 1** *GenESIRE(S)*

---

**Input:** A set of strings  $S$ **Output:** An *ESIRE*  $R$ 

```
1: Construct the  $G(V, E) = SOA(S)$  using 2T-INF [2];
2: Rename each node  $v$  with loop  $v^+$  and remove the loop; For each NTSCC, call algorithm Repair(NTSCC,  $S$ ). Then we get a new one  $G' = (V', E')$ ;
3:  $G.setln()$ ;  $R \leftarrow \varepsilon$ ;  $ln = 1$ ;
4: while  $ln \leq (ln \text{ of } G'.sink) - 1$  do
5:    $V_T \leftarrow$  all nodes with level number  $ln$  and  $length(sym(v)) \geq 2$ ;
6:    $V_S \leftarrow$  all nodes with level number  $ln$  and  $length(sym(v)) = 1$ ;
7:    $A \leftarrow Combine(V_S, "|")$ ;  $B \leftarrow Combine(V_T, "|")$ ;
8:   if  $A \neq \emptyset$  and  $B = \emptyset$  then
9:     if  $!G.isSL(ln)$  then
10:       $R \leftarrow R \cdot A$ ;
11:     else
12:       $R \leftarrow R \cdot A^?$ ;
13:     end if
14:   end if
15:   if  $A = \emptyset$  and  $B \neq \emptyset$  then
16:     if  $!G.isSL(ln)$  then
17:        $R \leftarrow R \cdot B$ ;
18:     else
19:        $R \leftarrow R \cdot B^?$ ;
20:     end if
21:   end if
22:   if  $A \neq \emptyset$  and  $B \neq \emptyset$  then
23:      $R \leftarrow R \cdot A^? \cdot B^?$ ;
24:   end if
25: end while
26: return  $R$ 
```

---

---

**Algorithm 2** *Repair(V, S)*

---

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

```
1:  $pattern \leftarrow V$ ;  $S' \leftarrow \bigcup_{s \in S} Filter(pattern, s)$ ; Compute sets  $CS(S')$ ,  $NCS(S')$  using POR( $S'$ );
2: if  $CS(S') = \emptyset$  then
3:   return  $(Graph(CS).combine(V))^+$ ;
4: else
5:    $G = Graph(CS)$ ;
6:   while  $G.nodes() \neq \emptyset$  do
7:      $v = clique\_removal(G)$ ;  $G = G \setminus v$ ;  $all\_mis.append(v)$ ;
8:   end while
9:   for each  $mis \in all\_mis$  do
10:     $sub\_ex = GenESIRE(\bigcup_{s \in S} Filter(mis, s))$ ;  $RE_{mis}.append((\varepsilon \in S')? sub\_ex^? : sub\_ex)$ ;
11:   end for
12:    $newRE \leftarrow Combine(RE_{mis}, "&")$ ;
13:   return  $newRE$ 
14: end if
```

---

1.  $V_S \neq \emptyset$ ,  $V_T = \emptyset$  and  $ln=1$  is not a skip level.

Let  $V_S = \{v_1, v_2, \dots, v_k\}$ .  $\alpha_1 = (v_1|v_2|\dots|v_k)$ . According to the algorithm, for each node  $v_i$  there is an edge connected with  $src$  and no edge between any two nodes  $v_i$  and  $v_j$ . Now we prove  $sym(\alpha_1) = sym(\beta_1)$ . If there is a symbol  $a \in sym(\alpha_1)$  but  $a \notin sym(\beta_1)$ , there exists some string  $s_0 \in S$  and  $s_0 \in L(\alpha)$  started with  $a$ . However,  $s_0 \notin L(\beta)$  which causes a contradiction with  $S \subseteq L(\beta)$ . If there is a symbol  $a \in sym(\beta_1)$  but  $a \notin sym(\alpha_1)$ , with similar analysis, it will lead to a contradiction with  $L(\beta) \subset L(\alpha)$ . Therefore,  $sym(\alpha_1) = sym(\beta_1)$  which can also hold on for the following cases.

If there are concatenation ( $\cdot$ ) or interleaving ( $\&$ ) operators in  $\beta_1$ , there must be edges between two nodes  $v_i, v_j$  to illustrate the occurrence orders. However, this will generate strings not in  $L(\alpha)$ , which is a contradiction with  $L(\beta) \subset L(\alpha)$ . For unary operators, each symbol in  $\beta_1$  must be the same with  $\alpha_1$ , otherwise it will lead to a contradiction with  $L(S) \subset L(\beta)$  or  $L(\beta) \subset L(\alpha)$ . Therefore, we have  $\alpha_1 = \beta_1$  and  $L(\alpha_1) = L(\beta_1)$ .

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

Let  $V_S = \{v_1, v_2, \dots, v_k\}$ .  $\alpha_1 = (v_1|v_2|\dots|v_k)^?$ . According to the algorithm *GenESIRE*, there must be edges for  $src$  to some node  $v \in V \setminus \{src\} \setminus V_S$ . Similar with the proof in case 1, we know that  $\beta_1$  must be added with optional operator  $?$ , otherwise  $L(\beta)$  can not cover the whole set  $S$  which is a contradiction. Therefore, we have  $\alpha_1 = \beta_1$  and  $L(\alpha_1) = L(\beta_1)$ .

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

Let  $V_T = \{A_1, A_2, \dots, A_k\}$  where  $A_i$  is the node consisting of multiple symbols.  $\alpha_1 = (A_1|A_2|\dots|A_k)$ . Similar with the proof in case 1, we know that there is an edge from  $src$  to node  $A_i$  and there is no edge between any node  $A_i$  and  $A_j$  where  $i, j \in [1, k]$ . Symbols of any two nodes  $sym(A_i)$  and  $sym(A_j)$  can not occur in one string. Therefore we know that  $\beta_1$  must be in the same form of  $\beta_1 = (B_1|B_2|\dots|B_k)$  in order to satisfy the condition  $L(S) \subseteq L(\beta) \subset L(\alpha)$ . Suppose that  $B_i$  in  $\beta_1$  is in some specific order, it is easy to conclude that  $sym(A_i) = sym(B_i)$ . Now we prove  $A_i = B_i$ .

According to the algorithm *Repair(S')*,  $A_i$  is a sequence of *ESs* connected with interleaving  $\&$ . Suppose that  $A_i = s_i^1 \& s_i^2 \& \dots \& s_i^q$  where  $s_i^p$  is an *ES* and  $p \in [1, q]$ . Each  $s_i^p$  is a maximum independent set obtained by *Graph(CS)*. Symbols within  $s_i^p$  are ordered determined by *NCS* while symbols between  $s_i^p$  and  $s_i^r$  are unordered. Both *CS* and *NCS* are computed from *POR(S')* ( $S' = \bigcup_{s \in S} Filter(sym(A_i), s)$ ). Remember  $\beta$  is also an *ESIRE*. Suppose any symbols  $a \in sym(s_i^p)$  and  $b \in sym(s_i^r)$ , if they were connected by concatenation operator instead of interleaving, only one partial order ( $a \prec b$  or  $b \prec a$ ) would appear in strings generated by  $L(\beta)$ . This is a contradiction with condition  $S \subseteq L(\beta)$ . If symbols  $a$  and  $b$  within  $s_i^p$  were connected by interleaving operator instead of concatenation or union, then partial orders  $a \prec b$  and  $b \prec a$  would both appear in strings in  $L(\beta)$  while can not in  $L(\alpha)$ . This causes another contradiction with  $L(\beta) \subset L(\alpha)$ . Therefore we can conclude that  $\beta_1$  is in the form of  $g_i^1 \& g_i^2 \& \dots \& g_i^q$  in which  $sym(s_i^p) = sym(g_i^p)$  ( $g_i^p$  in  $\beta_1$  is ordered in accordance with  $A_i$ ) and

symbols within  $g_i^p$  only have two kinds of binary operators: concatenation and union.

Next, the specific form of  $s_i^p$  is determined by calling the algorithm *GenESIRE* in which  $V_T = \emptyset$  and  $V_S \neq \emptyset$ . This is just the situation which has been proved in case 1 and case 2. We can easily concluded that  $s_i^p = g_i^p$ . Therefore, we have  $A_i = B_i$ ,  $\alpha_1 = \beta_1$  and  $L(\alpha_1) = L(\beta_1)$ .

4.  $V_S = \emptyset$ ,  $|V_T| \neq \emptyset$  and  $ln=1$  is a skip level.

Let  $V_T = \{A_1, A_2, \dots, A_k\}$  where  $A_i$  is the node consisting of multiple symbols.  $\alpha_1 = (A_1|A_2|\dots|A_k)^?$ . Similar with the proof in case 2, there exists an edge from  $src$  to node  $v \in V \setminus \{src\} \setminus V_T$ . In order to accept all strings in  $S$ ,  $\beta_1$  must have optional operator also. Therefore, we have  $\alpha_1 = \beta_1$  and  $L(\alpha_1) = L(\beta_1)$ .

5.  $V_S \neq \emptyset$ ,  $|V_T| \neq \emptyset$ .

According to the analysis above, all factors have optional operator in  $\alpha_1$ .  $\beta_1$  must be in the same form in order to satisfy the condition  $S \subseteq L(\beta) \subset L(\alpha)$ . Therefore,  $L(\alpha_1) = L(\beta_1)$ .

From the analysis above, we can conclude that  $L(\alpha_1) = L(\beta_1)$  holds. Then we consider  $\alpha' = \alpha_2 \dots \alpha_n$  and  $\beta' = \beta_2 \dots \beta_m$ . We first construct a new set of strings  $S'$ . For each string  $s \in S$ , if  $sym(s) \cap sym(\alpha_1) = \emptyset$ , then  $s \in S'$ ; otherwise, replace all alphabet in  $sym(\alpha_1)$  as  $\varepsilon$  and add it to  $S'$ . We let  $G_{S'} = SOA(S')$ . Clearly, we can find if we merge  $\alpha_1$  together with  $src$  ( $\alpha_1 \cdot src$ ) and consider it as the new  $src$ , then the new graph is the same with  $G_{S'}$  and  $\alpha' = GenESIRE(G_{S'})$ . We can use the same proof procedure as above until  $ln=n$ . We therefore can conclude that  $L(\alpha') = L(\beta')$  and  $L(\alpha) = L(\beta)$ .

## References

1. Boppana, R., Halldrsson, M.M.: Approximating Maximum Independent Set by Excluding Subgraphs. *Bit Numerical Mathematics* 32(2), 180–196 (1992)
2. Garcia, P., Vidal, E.: Inference of k-Testable Languages in the Strict Sense and Application to Syntactic Pattern Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(9), 920–925 (2002)
3. Peng, F., Chen, H.: Discovering Restricted Regular Expressions with Interleaving 9313, 104–115 (2015)