

8 APPENDIX

8.1 proof of Theorem 3.6

(1) The deterministic FAS recognizes the language defined by SOREFs.

PROOF. According to the definition of an FAS, an FAS is defined to recognize the language defined by a SOREF. For a SOREF $r = r_{i_1} \& r_{i_2} \& \dots \& r_{i_k}$ ($i, k \in \mathbb{N}, k \geq 2$), there are start marker $\&_i$ and end marker $\&_i^+$ in an FAS for recognizing the strings derived by r . For each subexpression r_{i_j} ($1 \leq j \leq k$) in r , there is a concurrent marker $\|_{ij}$ in an FAS for recognizing the symbols or strings derived by r_{i_j} .

In addition, for strings recognition, an FAS recognizes a string by treating symbols in a string individually. A symbol y in a string $s \in \mathcal{L}(r)$ (r is a SOREF) is recognized if and only if the current state (set) p is reached such that $y \in p$. A SOREF r is a deterministic expression, every symbol in s can be uniquely matched in r , and for every symbol l in r , there must exist a state (set) in an FAS including l . According to the transition function of an FAS, for the deterministic FAS \mathcal{A} , every symbol in s can be recognized in a state in \mathcal{A} . Therefore, $s \in \mathcal{L}(\mathcal{A})$. Then, $\mathcal{L}(r) \subseteq \mathcal{L}(\mathcal{A})$. The deterministic FAS recognizes the language defined by SOREFs. \square

(2) The membership problem for deterministic FAS is decidable in polynomial time. I.e., for any string s , and a deterministic FAS \mathcal{A} , we can decide whether $s \in \mathcal{L}(\mathcal{A})$ in polynomial time.

PROOF. An FAS recognizes a string by treating symbols in a string individually. A symbol y in a string s is recognized if and only if the current state p is reached such that $y \in p$. Let p_y denote the state (set) p including symbol (node) y . The next symbol of y is read if and only if y has been recognized at a state p_y . Let G denote the state-transition diagram of an FAS \mathcal{A} . The number of nodes in G is $\lceil \log_2 |\Sigma| \rceil + 2|\Sigma| + 2$ (including q_0 and q_f) at most. Assume that the current read symbol is y and the current state is q :

- (1) $|q| \geq 1, \exists v \in q : y \in G. \succ (v)$ ($v \in \{\|_{ij}\}_{i \in \mathbb{D}_\Sigma, j \in \mathbb{P}_\Sigma} \cup \Sigma$).
A state (set) q includes $\lceil \log_2 |\Sigma| \rceil + 2|\Sigma|$ nodes at most. For deterministic FAS, it takes $\mathcal{O}(|\Sigma|)$ time to search the node v . Then, the state $p_y = q \setminus \{v\} \cup \{y\}$ can be reached, y is recognized. Thus, for the current state q , it takes $\mathcal{O}(|\Sigma|)$ time to recognize y .
- (2) $|q| \geq 1, \exists \&_i \in q : y \in \mathcal{R}(\&_i)$.
For deterministic FAS, it takes $\mathcal{O}(|\Sigma|)$ time to search the node $\&_i$ in state (set) q , and it also takes $\mathcal{O}(|\Sigma|)$ time to decide whether $y \in \mathcal{R}(\&_i)$. Then, the state q transits to the state $q' = q \setminus \{\&_i\} \cup \{\|_{ij} \mid \|_{ij} \in G. \succ (\&_i), j \in \mathbb{P}_\Sigma\}$. Then, there is a node $\|_{ij}$ in q' that is checked whether $y \in G. \succ (\|_{ij})$. Case (1) will be considered. then, for the current state q , it takes $\mathcal{O}(|\Sigma|^2)$ time to recognize y .
- (3) $|q| \geq 1, \exists \&_i^+ \in q : y \in \mathcal{R}(\&_i)$
The state $\&_i^+$ will transit to the state $\&_i$, case (2) is satisfied. Then, for the current state q , it takes $\mathcal{O}(|\Sigma|^2)$ time to recognize y .
- (4) $q = q_0$.
If $y \in G. \succ (q_0)$, then, for deterministic FAS, it takes $\mathcal{O}(|\Sigma|)$ time to search the node y . Otherwise, a node

$\&_i$ ($i \in \mathbb{D}_\Sigma$) is searched and is decided whether $y \in \mathcal{R}(\&_i)$. Then, it takes $\mathcal{O}(|\Sigma|^2)$ time for q transiting to the state $\&_i$. Case (2) is satisfied. Then, for the current state q , it takes $\mathcal{O}(|\Sigma|^2)$ time at most to recognize y .

Thus, for deterministic FAS, and any read symbol y and a current state q , it takes $\mathcal{O}(|\Sigma|^2)$ time at most to recognize y . let $|s|$ denote the length of a string s , then for an FAS, it takes $\mathcal{O}(|s||\Sigma|^2)$ time to recognize s . Therefore, the membership problem for a deterministic FAS is decidable in polynomial time (uniform)¹³. \square

8.2 proof of Theorem 4.4

PROOF. $P_{<}(v_1, v_2) = P_{<}(v_2, v_1) = 1$, which precisely captures the shuffle of v_1 and v_2 . If there exists $v_1, v_2 \in \Sigma : P_{<}(v_1, v_2) = P_{<}(v_2, v_1) = 1$ and $(v_1, v_2) \notin U_{\&}$, then according to line 19 in algorithm 2. The tuple (v_1, v_2) is put in $U_{\&}$. $(v_1, v_2) \in U_{\&}$, this is a contradiction. Then, there does not exist $v_1, v_2 \in \Sigma : P_{<}(v_1, v_2) = P_{<}(v_2, v_1) = 1$ and $(v_1, v_2) \notin U_{\&}$. $U_{\&}$ precisely capture any two symbols that the shuffle of them is extractable from sample S . \square

8.3 proof of Theorem 4.7

PROOF. For any given finite sample S and a group g in $P_{\&}$, let $r' = r(e'_1) \& \dots \& r(e'_{k'})$ ($k' \geq 1$). Assume that there exists a group $g' = \{e'_1, \dots, e'_{k'}\}_{\&}$ such that $\mathcal{L}(r(e_1) \& \dots \& r(e_k)) \supset \mathcal{L}(r') \supseteq S$. Then, $k' \leq k$.

If $k' < k$, then the group g does not have the minimum number of the set of symbols. However, according to lines 5, 9~10, 7 and 13 in Algorithm 3, the group g with minimum size can be ensured. Then, $k' < k$ does not hold.

If $k' = k$, then $\mathcal{L}(r(e_1) \& \dots \& r(e_k)) \supset \mathcal{L}(r(e'_1) \& \dots \& r(e'_k)) \supseteq S$. This indicates that there exists e_i ($1 \leq i \leq k$) in group g that does not include as many symbols as possible. However, according to lines 2~3 in Algorithm 3, a node (symbol) v with maximum degree is identified, the successors of the node v form a set that can be put in a group, i.e., e_i in group g can include maximum number of symbols if g is updated that other nodes can also be put in e_i . There is a contradiction to the assumption. The conclusion in Theorem 4.7 holds. \square

8.4 Proof of Theorem 4.9

(1) The FAS \mathcal{A} is a deterministic FAS.

PROOF. The FAS \mathcal{A} is constructed from an SOA, which is a deterministic automaton that every node is labelled by distinct alphabet symbol. And each symbol y in a string, y is recognized if and only if a state (set) p including symbol (node) y is reached. Thus, each symbol y in a string can be deterministically recognized. The FAS \mathcal{A} is a deterministic FAS. \square

(2) There does not exist an FAS \mathcal{A}' , which is learnt from S such that $\mathcal{L}(\mathcal{A}) \supset \mathcal{L}(\mathcal{A}') \supseteq S$.

PROOF. Assume that there exists an FAS \mathcal{A}' learnt from S such that $\mathcal{L}(\mathcal{A}) \supset \mathcal{L}(\mathcal{A}') \supseteq S$. Then, according to the

¹³Note that, for non-uniform version of the membership problem for a deterministic FAS, only the string to be tested is considered as input. This indicates that $|\Sigma|$ is a constant. In this case, the membership problem for a deterministic FAS is decidable in linear time.

states specifications in an FAS, the states $\&_i$ and $\&_i^+$ ($i \in \mathbb{D}_\Sigma$) are markers for recognizing shuffled strings, which can be captured by $r_1 = r(e'_1)\&r(e'_2)\&\dots\&r(e'_{k'})$, where e'_i ($1 \leq i \leq k'$) is a set of symbols and $r(e'_i)$ is a regular expression such that $\Sigma_{r(e'_i)} = e'_i$. According to algorithm 4, the FAS \mathcal{A} is constructed from the SOA \mathcal{A} built for S by extracting sets of symbols (nodes) in each group in $P_\&$. The states $\&_i$ and $\&_i^+$ ($i \in \mathbb{D}_\Sigma$) are markers for recognizing shuffled strings, which can also be captured by $r_2 = r(e_1)\&r(e_2)\&\dots\&r(e_k)$ ($k \geq 1$). The sets of symbols (nodes) in each group are e_1, e_2, \dots, e_k .

According to Theorem 4.7, if both r_1 and r_2 recognize the shuffled strings extractable from S , then $\mathcal{L}(r_1) \supset \mathcal{L}(r_2) \supseteq S$. And the SOA \mathcal{A} built for S such that there does not exist an SOA \mathcal{B} such that $\mathcal{L}(\mathcal{B}) \supset \mathcal{L}(\mathcal{A}) \supseteq S$. This implies that the shuffled strings and non-shuffled strings are recognized by FAS \mathcal{A} , are also recognized by FAS \mathcal{A}' . Then, $\mathcal{L}(\mathcal{A}') \supseteq \mathcal{L}(\mathcal{A})$. There is a contradiction to the initial assumption. Therefore, there does not exist an FAS \mathcal{A}' , which is learnt from S such that $\mathcal{L}(\mathcal{A}) \supset \mathcal{L}(\mathcal{A}') \supseteq S$. \square

8.5 Proof of Theorem 5.2

(1) r is a SOREF.

PROOF. Assume that r derived by algorithm *InfSOREF* is not a SOREF. Algorithm *InfSOREF* mainly transforms the learnt FAS $\mathcal{L}(\mathcal{A})$ to r by using algorithm *Soa2Sore*. According to the definition of an FAS, every node labelled alphabet symbol occurs once in an FAS, and the algorithm *Soa2Sore* can transform the FAS $\mathcal{L}(\mathcal{A})$ to a SORE, every alphabet symbol in r occurs once. According to Theorem 27 presented in [19], there is $\mathcal{L}(r) \supseteq \mathcal{L}(\mathcal{A})$. Moreover, the learnt FAS is deterministic automaton, which recognizes the language defined by SOREFs. r also recognizes the language defined by SOREFs. Since every alphabet symbol in r occurs once, r is a SOREF. \square

(2) There does not exist a SOREF r' such that $\mathcal{L}(r) \supset \mathcal{L}(r') \supseteq \mathcal{L}(\mathcal{A})$.

PROOF. Assume that there exists a SOREF r' such that $\mathcal{L}(r) \supset \mathcal{L}(r') \supseteq \mathcal{L}(\mathcal{A})$. The FAS \mathcal{A} can be considered as an SOA. According to Theorem 27 presented in [19], a SORE r_s is transformed from the SOA \mathcal{A} by using algorithm *Soa2Sore*, there does not exist a SORE r'_s such that $\mathcal{L}(r_s) \supset \mathcal{L}(r'_s) \supseteq \mathcal{A}$. According to algorithm 7, r_s and r'_s can be rewritten to SOREFs r and r' (no loss of precision), respectively. For an FAS \mathcal{A} , there does not exist a SOREF r' such that $\mathcal{L}(r) \supset \mathcal{L}(r') \supseteq \mathcal{A}$. There is a contradiction to the initial assumption. Therefore, there does not exist a SOREF r' such that $\mathcal{L}(r) \supset \mathcal{L}(r') \supseteq \mathcal{L}(\mathcal{A})$. \square