8 APPENDIX

8.1 Proof of Theorem 3.4

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

Proof. According to the definition of an FAS, for a SOREF r, and the ith subexpression of the form $r_i = r_{i_1} \& r_{i_2} \& \cdots \& r_{i_k}$ $(i, k \in \mathbb{N}, k \geq 2)$ in r, there are start marker $\&_i$ and end marker $\&_i^+$ in an FAS for recognizing the strings derived by r_i. For each subexpression r_{i_j} $(1 \leq j \leq k)$ in r_i, 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 $s \in \mathcal{L}(r)$ is recognized if and only if the current state (a set of nodes) p is reached such that $y \in p$. The end symbol $\dashv$ is recognized if and only if the final state is reached. If y (resp. $\dashv$) is not consumed, then y (resp. $\dashv$) will be still read as the current symbol to be recognized. 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 (a set of nodes) 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 $\mathcal{A}$. When the last symbol of s was recognized, the end symbol $\dashv$ is read as the current symbol, suppose the current state is q, q will be finally transit to the state q_f such that $\dashv$ is consumed. 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. ◻

(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 $O(|s||\Sigma|^2)$ 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 (a set of nodes) p including symbol y. The next symbol of y is read if and only if y has been recognized at the state p_y. H is the node transition graph of an FAS $\mathcal{A}$. The number of nodes in H 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 is a set: $|q| \geq 1$ and $\exists v \in \{||_{ij}\}_{i \in \mathbb{D}_\Sigma, j \in \mathbb{P}_\Sigma} \cup \Sigma : v \in q \wedge y \in H.\succ(v)$.
A state (set) q includes $\lceil \log_2 |\Sigma| \rceil + 2|\Sigma|$ nodes at most. For deterministic FAS, it takes $O(|\Sigma|)$ time to search the node v. Then, the state $p_y = q \backslash \{v\} \cup \{y\}$ can be reached, y is recognized. Thus, for the current state q, it takes $O(|\Sigma|)$ time to recognize y.

(2) q is a set: $|q| \geq 1$ and $\exists \&_i \in q : y \in H.R(\&_i)$.
For deterministic FAS, it takes $O(|\Sigma|)$ time to search the node $\&_i$ in state (set) q, and it also takes $O(|\Sigma|)$ time to decide whether $y \in H.R(\&_i)$. Then, the state q transits to the state $q' = q \backslash \{\&_i\} \cup H.\succ(\&_i)$. Then, there is a node $||_{ij}$ $(||_{ij} \in H.\succ(\&_i), j \in \mathbb{P}_\Sigma\})$ in q' that is checked whether $y \in H.\succ(||_{ij})$. Case (1) will be considered. Then, for the current state q, it takes $O(|\Sigma|^2)$ time to recognize y.

(3) q is a set: $|q| \geq 1$ and $\exists \&_i^+ \in q : y \in H.R(\&_i)$.
The state including the node $\&_i^+$ will transit to the state including the node $\&_i$, case (2) is satisfied. Then, for the current state q, it takes $O(|\Sigma|^2)$ time to recognize y.

(4) $q = q_0$.
If $y \in H.\succ(q_0)$, then, for deterministic FAS, it takes $O(|\Sigma|)$ time to search the state including the node y. Otherwise, a node $\&_i$ $(i \in \mathbb{D}_\Sigma)$ is searched and then is decided whether $y \in H.R(\&_i)$. Then, it takes $O(|\Sigma|^2)$ time for q to transit to the state $\{\&_i\}$. Case (2) is satisfied. Then, for the current state q, it takes $O(|\Sigma|^2)$ time at most to recognize y.

Thus, for deterministic FAS, a symbol $y \in \Sigma_s$ and a current state q, it takes $O(|\Sigma|^2)$ time at most to recognize y. When the last symbol of s was recognized, the end symbol $\dashv$ requires to be consumed, it takes $O(|H.V|) = O(|\Sigma|)$ time to transit to the final state q_f. Let $|s|$ denote the length of the string s, then for an FAS, it takes $O(|s||\Sigma|^2)$ time to recognize s. Therefore, the membership problem for a deterministic FAS is decidable in polynomial time (uniform)[11]. ◻

8.2 Proof of Theorem 4.3

Proof. For each tuple $(u, v) \in U$ $(u \neq v)$, according to the definition of *necessary interleaving* (see Definition 2.5), algorithm *ShuffleTuples* ensures that there exists two strings $s_1, s_2 \in S$ such that $P(s_1, u, v) = P(s_2, v, u) = 1$ (lines 11∼12 and lines 14∼15). u can be interleaved with v, and there does not exist the two strings or substrings s and t occurring in S such that s and t can be described by $MutexStr(a, b)$ and $MutexStr(b, a)$, respectively (lines 19∼20). u is necessarily interleaved with v for S. Then, according to the definition of shuffle tuple (see Definition 2.7), (u, v) is a shuffle unit.

For any two distinct symbols $u, v \in \Sigma$, if u is necessarily interleaved with v for S, in algorithm *ShuffleTuples*, u and v will be searched from a SCC (lines 1∼3). According to the definition of *necessary interleaving*, we obtain the tuple (u, v) such that there exists two strings $s_1, s_2 \in S$ such that $P(s_1, u, v) = P(s_2, v, u) = 1$, (u, v) will be put into U (line 16). Moreover, there does not exist the two strings or substrings s and t occurring in S such that s and t can be described by $MutexStr(a, b)$ and $MutexStr(b, a)$, respectively, (u, v) will not be removed from U (lines 19∼20), then $(u, v) \in U$. ◻

8.3 Proof of Theorem 4.5

Proof. For a given undigraph $F(V, E)$, and a tuple $(u, v) \in U_\&$, because of $F.E = U_\&$, the node u connects the node v in F. Suppose that the current given undigraph is F, the current set of shuffle units is $P_\&$, and u has maximum node degree. The node u and the nodes (including the node v) connected to the node u compose two different sets e_1 and e_2 $(u \in e_1, v \in e_2)$, respectively. Let $\alpha = e_1 \cup e_2$.

In algorithm *ShuffleUnits*, if there exists a shuffle unit $l \in P_\&$ such that the intersection of the set $e \in l$ and the set $\beta \in \{e_1, e_2\}$ is not empty, either e $(e = e_1 \cup e_2)$ is partitioned (line 6) or the sets β and $\alpha \backslash \beta$ are merged with e and the union of the other sets from l, respectively (line 9), and the other sets from l are copied to form a new shuffle unit (lines 8, 10). Otherwise, add the shuffle unit $[e_1, e_2]$ into $P_\&$ (line 11).

Therefore, there always exits a shuffle unit in $P_\&$ such that u and v are in two distinct sets in the shuffle unit. Then, the node u and its associated edges are removed from F, algorithm *ShuffleUnits* recursively extract shuffle units from a new undigraph.

[11] Note 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.

Thus, there is a unique shuffle unit $l \in P_\&$ such that u and v are in distinct sets in l.

Assume that, there exists a set e_i ($i \in \mathbb{N}$) in a shuffle unit $l \in P_\&$ such that e_i can be partitioned into two new sets to form a new shuffle unit. Let $l = [e_1, \cdots, e_i, \cdots, e_k]$ ($k \geq 2, k \in \mathbb{N}$), e_i is partitioned into two sets e_i^1 and e_i^2 ($e_i = e_i^1 \cup e_i^2$), then the new shuffle unit $l' = [e_1, \cdots, e_i^1, e_i^2, \cdots, e_k]$. According to the definition of shuffle unit (see Definition 2.8), there the two elements $v_1 \in e_i^1$ and $v_2 \in e_i^2$ such that v_1 is necessarily interleaved with v_2 for S, then $(v_1, v_2) \in U_\&$ or $(v_2, v_1) \in U_\&$, there has been a unique shuffle unit $l_u \in P_\&$ such that v_1 and v_2 are in distinct sets in l_u. However, v_1 and v_2 are also in distinct sets in l'. There is a contradiction to the above conclusion. The assumption does not hold, then a set in a shuffle unit of $P_\&$ can not be partitioned into two sets, a shuffle unit in $P_\&$ has minimum size. □

8.4 Proof of Theorem 4.7

(1) The learned FAS $\mathcal{A}$ from a sample S is a deterministic FAS..

Proof. H is the node transition graph of an FAS $\mathcal{A}$. The FAS $\mathcal{A}$ is learned by constructing the node transition graph H. We convert the SOA G built for S to the digraph H. For the different markers $\&_i$, $\&_i^+$ and $||_{ij}$, where $i \in \mathbb{D}_\Sigma$ and $j \in \mathbb{P}_\Sigma$, they are respectively added into the SOA G by traversing the shuffle units in $P_\&$. For different shuffle units in $P_\&$, there are different start markers $\&_i$ and end markers $\&_i^+$ which are added into G. For different sets (disjoint) in a shuffle unit, there are different concurrent markers $||_{ij}$ which are added into G. The finally obtained G is the node transition graph of the learned FAS $\mathcal{A}$. Then, every node of H is labelled by distinct symbol. This implies that, a state (a set of nodes) does not include the nodes labelled by the same symbol.

For recognizing a string $s \in S$, according to the state transition function of the learned FAS $\mathcal{A}$, a symbol $y \in \Sigma_s$ (resp. $\dashv$) is recognized if and only if the state (set) p including the node y (resp. the final state q_f) is reached. If y (resp. $\dashv$) does not been consumed, there is only one next state p' is specified that the state p' including the node which can reach to the node y (resp. the node q_f) in H. Thus, each symbol $a \in \Sigma_s \cup \{\dashv\}$ can be unambiguously recognized. The FAS $\mathcal{A}$ is a deterministic FAS. □

Lemma 8.1. *Let $P_\& = \{[e_1, e_2, \cdots, e_k]\}$ ($k \geq 2$), and let $r(e_i)$ ($1 \leq i \leq k$) denote a regular expression such that $e_i = \Sigma_{r(e_i)}$. Assume that the set $P_\&$ of shuffle units is returned by algorithm ShuffleUnits. For a given finite sample S, and a shuffle unit $l' = [e'_1, e'_2, \cdots, e'_t]$ ($t \geq 2$), if there exists $r(e_i)$ and $r'(e'_j)$ ($1 \leq j \leq t$): $\mathcal{L}(r(e_1)\& \cdots \& r(e_k)) \supset \mathcal{L}(r'(e'_1)\& \cdots \& r'(e'_t)) \supseteq S$, then $t = k$ and $e_i = e'_i$.*

Proof. For $\mathcal{L}(r(e_1)\& \cdots \& r(e_k)) \supset \mathcal{L}(r'(e'_1)\& \cdots \& r'(e'_t)) \supseteq S$[12], there is $t \leq k$.

For any two distinct symbols $u, v \in \Sigma$, if u is necessarily interleaved with v for S. u and v are in two distinct sets in l'. Otherwise, it will lead to $\mathcal{L}(r'(e'_1)\& \cdots \& r'(e'_t)) \not\supseteq S$. Then, according to Theorem 4.5, a shuffle unit in the set $P_\&$ returned by algorithm *ShuffleUnits* has minimum size, then $k \leq t$. Thus, there is $t = k$.

Let $\mathcal{L}(r(e_1)\& \cdots \& r(e_k)) \supset \mathcal{L}(r'(e'_1)\& \cdots \& r'(e'_k)) \supseteq S$. If there exists $1 \leq i \leq k$ such that $e_i \neq e'_i$, then $r(e_i) \neq r'(e'_i)$, there exists a string s' such that $s' \in \mathcal{L}(r'(e'_1)\& \cdots \& r'(e'_k))$ but $s' \notin \mathcal{L}(r(e_1)\& \cdots \& r(e_k))$. Then, $\mathcal{L}(r(e_1)\& \cdots \& r(e_k)) \not\supset \mathcal{L}(r'(e'_1)\& \cdots \& r'(e'_k))$. Therefore, $e_i = e'_i$ for any $1 \leq i \leq k$. □

[12] For simplicity of proof, let $r(e_1)\& \cdots \& r(e_k)$ denote that there exists $r(e_i)$ such that $r(e_1)\& \cdots \& r(e_k)$ is a regular expression supporting shuffle.

(2) There does not exist an FAS $\mathcal{A}'$, which is learned from S such that $\mathcal{L}(\mathcal{A}) \supset \mathcal{L}(\mathcal{A}') \supseteq S$. The FAS $\mathcal{A}$ is a precise representation of S.

Proof. The FAS $\mathcal{A}$ is learned by constructing the corresponding node transition graph H. We convert the SOA G built for S to the digraph H by traversing shuffle units in $P_\&$, which is obtained from Algorithm 3. The built SOA G is a precise representation of S [18].

Assume that there exists an FAS $\mathcal{A}'$ learned from S such that $\mathcal{L}(\mathcal{A}) \supset \mathcal{L}(\mathcal{A}') \supseteq S$. For the node transition graph H' of the FAS $\mathcal{A}'$, H' should be constructed from the SOA G built for S, otherwise, the above assumption can not hold. Suppose that there is the set $P'_\&$ of shuffle units such that the digraph H' can be constructed from the SOA G by traversing shuffle units in $P'_\&$.

For each shuffle unit $l \in P_\&$, let $l = [e_1, \cdots, e_k]$ ($k \geq 2$), according to Algorithm 4, there are corresponding start marker $\&_m$ and end marker $\&_m^+$ ($m \in \mathbb{D}_\Sigma$) are added into G. Let $\mathcal{B}$ denote the constructed FAS. The FAS $\mathcal{B}$ can recognize the shuffled strings which consist of the symbols in $\bigcup_{1 \leq i \leq k} e_i$. Let $S_\&$ denote the set of the above shuffled strings extracted from S.

Then, for constructing FAS $\mathcal{A}'$, for each shuffle unit $l' \in P'_\&$ and $l' = [e'_1, \cdots, e'_t]$ ($t \geq 2$), we obtain the currently constructed FAS $\mathcal{B}'$ by adding the corresponding start marker $\&_n$ and end marker $\&_n^+$ ($n \in \mathbb{D}_\Sigma$) into G. If $\mathcal{L}(\mathcal{B}) \supset \mathcal{L}(\mathcal{B}') \supseteq S_\&$, then there exists $r(e_i)$ and $r'(e'_j)$ ($1 \leq j \leq t$) such that $\mathcal{L}(r(e_1)\& \cdots \& r(e_k)) \supset \mathcal{L}(r'(e'_1)\& \cdots \& r'(e'_t)) \supseteq S_\&$ (Let $\mathcal{L}(\mathcal{B}) = \mathcal{L}(r(e_1)\& \cdots \& r(e_k))$ and $\mathcal{L}(\mathcal{B}') = \mathcal{L}(r'(e'_1)\& \cdots \& r'(e'_t))$.). According to Lemma 8.1, there are $t = k$ and $e_i = e'_i$, then there is $l = l'$.

This implies that, if $\mathcal{L}(\mathcal{A}) \supset \mathcal{L}(\mathcal{A}') \supseteq S$, there is $P_\& = P'_\&$. For digraphs H and H', they are both constructed from the SOA G, then there is $\mathcal{A} = \mathcal{A}'$ and $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}') \supseteq S$. There is a contraction to the initial assumption. Therefore, the initial assumption does not hold, the FAS $\mathcal{A}$ is a precise representation of S. □

8.5 Proof of Theorem 5.2

(1) r is a SOREF..

Proof. H is the node transition graph of the learned FAS. Algorithm *InfSOREF* mainly transforms the constructed digraph H to r by using algorithm *Soa2Sore*. According to the definition of an FAS, every symbol labels a node of H at most once. H is also an SOA if we respect markers ($\&_i$, $\&_i^+$ and $||_{ij}$, $i \in \mathbb{D}_\Sigma$, $j \in \mathbb{P}_\Sigma$) as alphabet symbols, and the algorithm *Soa2Sore* transforms the digraph H to a SORE r_s. r is obtained by introducing shuffle operators into r_s, and every alphabet symbol in r occurs once. r is a SOREF. □

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

Proof. Assume that there exists a SOREF r' such that $\mathcal{L}(r) \supset \mathcal{L}(r') \supseteq S$. Theorem 4.7 demonstrates that the FAS $\mathcal{A}$ returned by algorithm *LearnFAS* is a precise representation of S. Then, there is $\mathcal{L}(r) \supset \mathcal{L}(r') \supseteq \mathcal{L}(\mathcal{A}) \supseteq S$, otherwise, for the SOREF r', there is an equivalent FAS $\mathcal{A}'$ such that $\mathcal{A}$ is not a precise representation of S ($\mathcal{L}(\mathcal{A}') \supset \mathcal{L}(\mathcal{A}) \supseteq S$).

Additionally, the node transition graph H of the learned FAS $\mathcal{A}$ can be considered as an SOA if we respect markers $\&_i$, $\&_i^+$ and $||_{ij}$ ($i \in \mathbb{D}_\Sigma$, $j \in \mathbb{P}_\Sigma$) as alphabet symbols. Then, for recognizing the language denoted by H, we can replace the ith ($i \in \mathbb{N}$) subexpression rb_i of form $r1\& \cdots \& r_k$ ($k \geq 2$) in r (resp. r') with $rs_i = (||_{i1} r_1 | ||_{i2} r_2 | \cdots | ||_{ik} r_k)$, we replace rb_i with rs_i for each

$i \in \mathbb{N}$ (the reverse process with respect to lines 3~5 in algorithm *InfSOREF*), finally we obtain the expression r_s (resp. r'_s). Then, there is $\mathcal{L}(r_s) \supset \mathcal{L}(r'_s) \supseteq \mathcal{L}(H)$.

However, $r_s = Soa2Sore(\mathcal{A}.H)$, r_s can be regarded as a SORE. According to Theorem 27 presented in [18], a SORE r_s is transformed from the digraph H by using algorithm *Soa2Sore*, there does not exist a SORE r'_s such that $\mathcal{L}(r_s) \supset \mathcal{L}(r'_s) \supseteq \mathcal{L}(H)$. There is a contradiction. The initial assumption does not hold. There does not exist a SOREF r' such that $\mathcal{L}(r) \supset \mathcal{L}(r') \supseteq S$. r is a precise representation of any given finite sample. □

8.6 Proof of Theorem 5.3

Proof. According to Theorem 3.4, an FAS can recognize the language defined by SOREFs. This implies that, for any given SOREF r, an equivalent FAS $\mathcal{A}$ can be constructed from the SOREF r. There must exist a finite sample S derived by r such that $\mathcal{A} = LearnFAS(S)$ ($\mathcal{L}(\mathcal{A}) \supseteq S$). The FAS $\mathcal{A}$ is transformed to a SOREF r' by using algorithm *InfSOREF*. According to Theorem 5.2, algorithm *InfSOREF* returns a SOREF which is a precise representation of S. Thus, $\mathcal{L}(r') = \mathcal{L}(\mathcal{A}) = \mathcal{L}(r) \supseteq S$.

Note that, r' and r are SOREFs where every alphabet symbol occurs at most once. r' does not comprise the redundant operators that is ensured by subroutine *Soa2Sore* [18] and lines 3~5 in algorithm *InfSOREF*. For the redundant operators and regular expression e occurring in r[13] , if there is a simplified regular expression e': $\Sigma_e = \Sigma_{e'}$ and $\mathcal{L}(e) = \mathcal{L}(e')$, then $e = e'$ (such as $(e^+)^+ = e^+$, $(e?)? = e?$, $((e^+)?)^+ = (e^+)?$, etc.) [4, 6, 18]. Therefore, for any given SOREF r, there exists a finite sample S such that r=*InfSOREF*(S). □

[13]Usually, the SOREF r provided by users does not comprise the redundant operations.