8 Appendix

8.1 Proof of Theorem 1

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

Proof. According to the definition of an FAS, for a SOREF r, 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 (set) 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 (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 $\mathcal{A}$. 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 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 (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. $|q| \geq 1$, $\exists v \in q : y \in H.\succ(v)$ $(v \in \{||_{ij}\}_{i \in \mathbb{D}_\Sigma, j \in \mathbb{P}_\Sigma} \cup \Sigma)$.
   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 $p_y = q \backslash \{v\} \cup \{y\}$ can be reached, y is recognized. Thus, for the current state q, it takes $\mathcal{O}(|\Sigma|)$ time to recognize y.
2. $|q| \geq 1$, $\exists \&_i \in q : y \in H.R(\&_i)$.
   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 $y \in H.R(\&_i)$. 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 $\mathcal{O}(|\Sigma|^2)$ time to recognize y.

3. $|q|\geq 1$, $\exists \&_i^+ \in q : y \in H.R(\&_i)$
   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 $y \in H.\succ(q_0)$, 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 $y \in H.R(\&_i)$. 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, a symbol $y \in \Sigma_s$ and a current state q, it takes $\mathcal{O}(|\Sigma|^2)$ time at most to recognize y. When the last symbol of s was recognized, the end symbol $\dashv$ requires to be consumed, it takes $\mathcal{O}(|H.V|) = \mathcal{O}(|\Sigma|)$ time to transit to the final state q_f. Let $|s|$ denote the length of the string s, then for an FAS, it takes $\mathcal{O}(|s||\Sigma|^2)$ time to recognize s. Therefore, the membership problem for a deterministic FAS is decidable in polynomial time (uniform)[7].

8.2 Proof of Theorem 2

(1) The learnt 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 learnt by constructing the node transition graph H. We convert the SOA G built for S to the digraph H, and the different markers ($\&_i$, $\&_i^+$, $||_{ij}$, $i \in \mathbb{D}_\Sigma, \in \mathbb{P}_\Sigma$) are 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 learnt FAS $\mathcal{A}$. Then, every node of H is labelled by distinct symbol.

For recognizing a string $s \in S$, according to the state transition function of the learnt FAS $\mathcal{A}$, a symbol $y \in \Sigma_s$ (resp. $\dashv$) is recognized if and only if the state (set) p including 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 node y (resp. node q_f) in H. Thus, each symbol $y' \in \Sigma_s \cup \{\dashv\}$ can be unambiguously recognized. The FAS $\mathcal{A}$ is a deterministic FAS.

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

[7] 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.

Proof. Assume that there exists a shuffle unit $l' = [e'_1, e'_2, \cdots, e'_t]$ $(t \geq 2)$ such that $\mathcal{L}(r(e_1)\&\cdots\&r(e_k)) \supset \mathcal{L}(r(e'_1)\&\cdots\&r(e'_t)) \supseteq S$. Then, $t \leq k$.

Let $l = [e_1, e_2, \cdots, e_k]$. There is $\Sigma = \bigcup_{1\leq i\leq k} e_i = \bigcup_{1\leq i\leq t} e'_i$. According to Algorithm 2, the returned $P_\& = \{[e_1, e_2, \cdots, e_k]\}$ implies that any two distinct symbols u and v in one set in l satisfies that u is unnecessarily interleaved with v for S. If $t < k$, there must exist two distinct symbols u' and v' are in the same set in l' such that u' is necessarily interleaved with v' for S. However, u is unnecessarily interleaved with v for language $\mathcal{L}(r(e'_1)\&\cdots\&r(e'_t))$. This will result in $\mathcal{L}(r(e'_1)\&\cdots\&r(e'_t)) \not\supseteq S$. Thus, $t = k$.

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 $1 \leq i \leq k$. There is a contradiction to the initial assumption. Thus, there does not exist a shuffle unit $[e'_1, e'_2, \cdots, e'_t]$ $(t \geq 2)$ such that $\mathcal{L}(r(e_1)\&\cdots\&r(e_k)) \supset \mathcal{L}(r(e'_1)\&\cdots\&r(e'_t)) \supseteq S$.

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

Proof. The FAS $\mathcal{A}$ is learnt by constructing the node transition graph H of the FAS. We convert the SOA G built for S to the digraph H by traversing shuffle units in $P_\&$, which is obtained from Algorithm 2. The built SOA G is a precise representation of S [11].

Assume that there exists an FAS $\mathcal{A}'$ learnt 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 by using 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' is 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 3, there are corresponding start marker $\&_i$ and end marker $\&_i^+$ are added into G. Let the current digraph G denote the node transition graph of an FAS, and $\mathcal{B}$ denote the 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, $\mathcal{L}(r(e_1)\&\cdots\&r(e_k)) \supseteq S_\&$. According to Lemma 1, there does not exist a shuffle unit $l' = [e'_1, e'_2, \cdots, e'_t] \in P'_\&$ $(t \geq 2)$ such that $\mathcal{L}(r(e_1)\&\cdots\&r(e_k)) \supset \mathcal{L}(r(e'_1)\&\cdots\&r(e'_t)) \supseteq S_\&$. This implies that, for each shuffle unit l' in $P'_\&$, the corresponding start marker $\&_k$ and end marker $\&_k^+$ $(k \in \mathbb{D}_\Sigma)$ are added into G to form the corresponding FAS $\mathcal{B}'$, $\mathcal{L}(\mathcal{B}) \supset \mathcal{L}(\mathcal{B}') \supseteq S_\&$ can not hold for $\mathcal{L}(\mathcal{B}) = \mathcal{L}(r(e_1)\&\cdots\&r(e_k))$ and $\mathcal{L}(\mathcal{B}') = \mathcal{L}(r(e'_1)\&\cdots\&r(e'_t))$.

Then, there does not exist the set $P'_\&$ of shuffle units such that the digraph H' is constructed from the SOA G by traversing shuffle units in $P'_\&$, and then $\mathcal{L}(\mathcal{A}) \supset \mathcal{L}(\mathcal{A}') \supseteq S$. Therefore, the initial assumption does not hold. The FAS $\mathcal{A}$ is a precise representation of S.

8.3 Proof of Theorem 3

(1) r is a SOREF.

Proof. H is the node transition graph of the learnt 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 $\mathcal{L}(r) \supset \mathcal{L}(r') \supseteq \mathcal{L}(\mathcal{A})$. The node transition graph H of the learnt FAS $\mathcal{A}$ can be considered as an SOA. According to Theorem 27 presented in [11], a SORE r_s is transformed from the digraph H by using algorithm $Soa2Sore$, there does not exist a SORE r'_s such that $\mathcal{L}(r_s) \supset \mathcal{L}(r'_s) \supseteq \mathcal{L}(H)$. According to algorithm 4, 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 $\mathcal{L}(r) \supset \mathcal{L}(r') \supseteq \mathcal{L}(\mathcal{A})$. There is a contradiction to the initial assumption. Therefore, there does not exist a SOREF r' such that $\mathcal{L}(r) \supset \mathcal{L}(r') \supseteq \mathcal{L}(\mathcal{A})$. Note that, $\mathcal{L}(r) \supseteq \mathcal{L}(\mathcal{A}) \supseteq S$ holds by Theorem 2. And Corollary 17 [11] implies that, a precise SOREF r (for any given finite sample) satisfies that $\mathcal{L}(r) \supseteq \mathcal{L}(\mathcal{A})$. There also 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.

8.4 Proof of Theorem 4

Proof. According to Theorem 1, 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 3, algorithm *InfSOREF* returns a SOREF which is a precise representation of S. Thus, $\mathcal{L}(r') = \mathcal{L}(\mathcal{A}) = \mathcal{L}(r) \supseteq S$. Therefore, for any given SOREF r, there exists a finite sample S such that $r = \mathit{InfSOREF}(S)$.