# 8 Appendix

## 8.1 proof of Theorem 1

(1) 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 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 $p_y = q \backslash \{v\} \cup \{y\}$ can be reached, $y$ is recognized. Thus, for the current state $q$, it takes $\mathcal{O}(|\Sigma|)$ time to recognize $y$.
2. $|q| \geq 1$, $\exists \&_i \in q : y \in \mathcal{R}(\&_i)$.
   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 $q' = q \backslash \{\&_i\} \cup \{\, ||_{ij} \,|\, ||_{ij} \in G.\succ(\&_i), j \in \mathbb{P}_\Sigma\}$. 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)[10].

[10]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.

(2) The deterministic FAS $\mathcal{A}$ 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 the $i$th 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 a SOREF $r$, 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 $s\in\mathcal{L}(r)$ ($r$ is a SOREF) is recognized if and only if the current state (a set of nodes) $p$ 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.

### 8.2 proof of Theorem 2

*Proof.* For any given finite sample $S$ and a group $g$ in $P_{\&}$, assume that there exists a SOREF $r'$ such that $\mathcal{L}(r(e_1)\&\cdots\&\ r(e_k))\supset\mathcal{L}(r')\supseteq S$. To ensure that $r'$ can capture $S$ (a finite set of shuffled strings), let $r'=r(e'_1)\&\cdots\&r(e'_{k'})$ $(k'\geq 1)$, let group $g'=[e'_1,\cdots,e'_{k'}]$. 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 $\mathcal{L}(r(e_1)\&\cdots\&r(e_k))\supset\mathcal{L}(r(e'_1)\&\cdots\&r(e'_k))\supseteq S$. 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 $v$ 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. There is a contradiction to the assumption. The conclusion in Theorem 2 holds.

### 8.3 Proof of Theorem 3

(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 (a set of nodes) $p$ including symbol $y$ is reached. Thus, each symbol $y$ in a string can be deterministically recognized. The FAS $\mathcal{A}$ is a deterministic FAS.

(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 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 $r_1 = r(e'_1)\&r(e'_2)\& \cdots \&r(e'_{k'})$, 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 $\mathscr{A}$ built for $S$ by extracting sets of symbols 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 $r_2 = r(e_1)\&r(e_2)\& \cdots \&r(e_k)$ $(k \geq 1)$. The sets of symbols in each group are $e_1$, $e_2$, $\cdots$, $e_k$.

According to Theorem 2, 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 $\mathscr{A}$ built for $S$ such that there does not exist an SOA $\mathscr{B}$ such that $\mathcal{L}(\mathscr{B}) \supset \mathcal{L}(\mathscr{A}) \supseteq S$. 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 $\mathcal{L}(\mathcal{A}) \supset \mathcal{L}(\mathcal{A}') \supseteq S$.

### 8.4 Proof of Theorem 4

(1) $r$ is a SOREF.

*Proof.* Assume that $r$ derived by algorithm *InfSOREF* is not a SOREF. Algorithm *InfSOREF* mainly transforms the learnt FAS $\mathcal{L}(A)$ 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 $\mathcal{L}(A)$ to a SORE, every alphabet symbol in $r$ occurs once. According to Theorem 27 presented in [14], there is $\mathcal{L}(r) \supseteq \mathcal{L}(A)$. 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.

(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 [14], 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 6, $r_s$ 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 $\mathcal{L}(r) \supset \mathcal{L}(r') \supseteq \mathcal{L}(\mathcal{A})$. Note that, $\mathcal{L}(r) \supseteq \mathcal{L}(\mathcal{A}) \supseteq S$ holds by Theorem 3.

### 8.5 Proof of Theorem 5

*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 $S$ obtained by traversing the FAS$\mathcal{A}$ such that $\mathcal{A} = \mathit{LearnFAS}(S)$ ($\mathcal{L}(A) \supseteq S$). The FAS $\mathcal{A}$ is transformed to a SOREF $r'$ by using algorithm *FAS2SOREF*. According to Theorem 4, algorithm *InfSOREF* returns a precise representation of $S$. Thus, $\mathcal{L}(r') = \mathcal{L}(\mathcal{A}) = \mathcal{L}(r)$. If $r' \neq r$, we can adjust the sample $S$ (adding special strings) such that the derived $r' = r$. Therefore, for any given SOREF $r$, there exists a finite sample $S$ such that $r = \mathit{InfSOREF}(S)$.