A Appendix

A.1 Proofs of Theorem 1

(1) An FAUC recognizes the language defined by a SOREUC.

Proof. According to the definition of an FAUC, for a SOREUC 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 is an unorder marker $\%_i^+$ in an FAUC 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 FAUC for recognizing the symbols or strings derived by r_{i_j}.

In addition, for strings recognition, an FAUC 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 SOREUC 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 FAUC including l. According to the transition function of an FAUC, there exists an FAUC $\mathcal{A}$ such that 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 finally transit to the state q_f such that $\dashv$ is consumed. Therefore, $s \in \mathcal{L}(\mathcal{A})$. Then, $\mathcal{L}(r) \subseteq \mathcal{L}(\mathcal{A})$. An FAUC recognizes the language defined by a SOREUC.

(2) The membership problem for FAUC is decidable in polynomial time. I.e., for any string s, and an FAUC $\mathcal{A}$, we can decide whether $s \in \mathcal{L}(\mathcal{A})$ in $\mathcal{O}(|s||\Sigma|^3)$ time.

Proof. An FAUC 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 FAUC $\mathcal{A}$. The number of nodes in H is $4|\Sigma|$ (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 $4|\Sigma|$ nodes at most. For an FAUC, it takes $\mathcal{O}(|\Sigma|^2)$ time to search the node v, where $y \in H.\succ(v)$. Then, the state $p_y = q \backslash \{v\} \cup \{y\}$ can be reached, y is recognized. Thus, for the current state q, it takes $\mathcal{O}(|\Sigma|^2)$ time to recognize y.
2. q is a set: $|q| \geq 1$ and $\exists v \in \{||_{ij}\}_{i \in \mathbb{D}_\Sigma, j \in \mathbb{P}_\Sigma}$: $v \in q \wedge ((\exists \%_t^+ \in H.\succ(v) \wedge y \in R(\%_t^+)) \vee (\exists +_k \in H.\succ(v) \wedge y \in R(+_k)))(t \in \mathbb{D}_\Sigma, k \in \mathbb{B}_\Sigma)$.
   For the current state q, it takes $\mathcal{O}(|\Sigma|)$ time to search the node v. If $+_k \in H.\succ(v)$ and $y \in R(+_k)$, since it needs $\mathcal{O}(|\Sigma|)$ time to decide whether $y \in R(+_k)$, it takes $\mathcal{O}(|\Sigma|^2)$ time at most to reach the state including the node y (i.e.,

recognizing y). If $\%_t^+ \in H.\succ(v)$ and $y \in R(\%_t^+)$, case (3) will be considered, it takes $\mathcal{O}(|\Sigma|^3)$ time at most to reach the state including the node y for another unorder marker can be a successor of $\%_t^+$.

3. q is a set: $|q| \geq 1$ and $\exists \%_i^+ \in q : y \in H.R(\%_i^+)$.
   For an FAUC, 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) and case (2) will be considered. Then, for the current state q, it takes $\mathcal{O}(|\Sigma|^3)$ time to recognize y.
4. $q = q_0$.
   If $y \in H.\succ(q_0)$, then, for an FAUC, it takes $\mathcal{O}(|\Sigma|)$ time to search the state including the node y. Otherwise, a node $\%_i^+$ ($i \in \mathbb{D}_\Sigma$) or a node $+_k$ ($k \in \mathbb{B}_\Sigma$) is searched and then is decided whether $y \in H.R(\%_i^+)$ or $y \in H.R(+_k)$. If the node $+_k$ is searched, it needs $\mathcal{O}(|\Sigma|)$ time to decide whether $y \in R(+_k)$, then it takes $\mathcal{O}(|\Sigma|)$ time at most to recognize y. If the node $\%_t^+$ is searched, then, case (3) and case (2) are considered, for the current state q, it takes $\mathcal{O}(|\Sigma|^3)$ time at most to recognize y.

Thus, for an FAUC, a symbol $y \in \Sigma_s$ and a current state q, it takes $\mathcal{O}(|\Sigma|^3)$ 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 FAUC, it takes $\mathcal{O}(|s||\Sigma|^3)$ time to recognize s. Therefore, the membership problem for an FAUC is decidable in polynomial time (uniform)[4].

A.2 Proof of Theorem 2

Proof. For any tuple $(a, b) \in U_\%$, the node a connects with the node b in the undigraph $F(V, E)$ ($F.E = U_\%$). The nodes a and b are in a connected component of F. According to the algorithm *UnorderUnits*, for each connected component f of F, there is a corresponding unorder unit.

First, the non-adjacent nodes, which are selected from f, compose a set M_f such that the sum of all node degrees is maximum. M_f is one of the sets in an unorder unit. Then if one of the nodes a and b occurs in M_f (a and b cannot occur in M_f at the same time), after removing the nodes in f and their associated edges, a new undigraph f' is obtained. If f' is not a connected graph, $[M_f, f'.V]$ forms an unorder unit, the other node occurs in $f'.V$. Otherwise, M_f is stored in an unorder unit, algorithm *UnorderUnits* recursively works on f', the other node must occur in another obtained set.

If neither a nor b occurs in M_f, after removing the nodes in f and their associated edges, a new undigraph f' is obtained, algorithm *UnorderUnits* recursively works on f'. In extreme case, $f'.V = \{a, b\}$ and $f'.E = \{(a, b)\}$, then

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

$M_{f'} = \{a\}$, $[\{a\}, \{b\}]$ forms an unorder unit. The nodes a and b occur in different sets.

All obtained unorder units are put into $P_{\%}$, thus, for any tuple $(a,b) \in U_{\%}$, there exists an unorder unit $ut \in P_{\%}$ such that a and b are in different sets in ut.

A.3 Proof of Theorem 3

Theorem 5. *For a finite sample S, let SOA $G = 2T\text{-}INF(S)$ and $P_{\%}$ denote the result returned by UnorderUnits, let $\mathcal{A} = ConsFauc(G, P_{\%})$, then $\mathcal{L}(\mathcal{A}) \supseteq S$.*

Proof. For a finite sample S, first, any two distinct alphabet symbols a and b which can be an unorder word for S are identified from S. $U_{\%}$ is the set of all tuples (a,b) identified from S. $P_{\%}$ is obtained by using algorithm *UnorderUnits* to recursively extract unorder units from the undigraph $F(V,E)$ where $F.E = U_{\%}$. Since an SOA built for S is a precise representation of S [13], and an unorder unit in $P_{\%}$ can be used to determine the substructure of an FAUC recognizing unordered strings, The SOA built for S is converted to the FAUC $\mathcal{A}$ by traversing the unorder units in $P_{\%}$. Theorem 2 ensures the constructed FAUC $\mathcal{A}$ can recognize all the unordered strings from \$S\$. The SOA G is built from S, $\mathcal{L}(\mathcal{A}) \supseteq S$, for any string $s \in S$ and any symbol a occurring in s, there is a node labelled by a in G. Then, there is also a node labelled by a in the node transition graph H of the FAUC $\mathcal{A}$.

Suppose that the current state is q (a set of nodes) and the current symbol a is read. According to the transition function of the constructed FAUC $\mathcal{A}$, if there is node $v \in q$ such that $a \in H.\succ(v)$, the state q will transit to the state including the node a (i.e., the state $q \setminus \{v\} \cup \{a\}$), a is recognized. If a is the first letter of the unordered string from s, the state q will transit to the state including the node $\%_i^+$ $(i \in \mathbb{D}_{\Sigma})$ such that the state including the node a can be reached via the node $\%_i^+$ $(a \in R(\%_i^+))$. If a is the first letter of the substring which repeatedly occurs in a string in S, the state q will transit to the state including the node $+_j$ $(j \in \mathbb{B}_{\Sigma})$ such that the state including the node a can be reached via the node $+_j$ $(a \in R(+_j))$. a is recognized if and only if the state including the node a is reached.

Thus, for any string $s \in S$ and any symbol a occurring in s, a can be recognized in the constructed FAUC $\mathcal{A}$, $s \in \mathcal{L}(\mathcal{A})$, then $\mathcal{L}(\mathcal{A}) \supseteq S$.

A.4 Proof of Theorem 4

Theorem 6. *For a finite sample S, let $r = InfSoreuc(S)$, then r is a SOREUC and $\mathcal{L}(r) \supseteq S$.*

Proof. For a finite sample S, the constructed FAUC $\mathcal{A}$ is returned by the algorithm *ConsFauc*, which is as a subroutine of algorithm *InfSoreuc*, Theorem 3 demonstrates that $\mathcal{L}(\mathcal{A}) \supseteq S$. After $Running(\mathcal{A}, S)$ return *true*, $\mathcal{L}(\mathcal{A}) \supseteq S$ is still hold.

In algorithm *GenSoreuc*, the node transition graph H of the FAUC $\mathcal{A}$ is first converted to a regular expression r_s by using the algorithm *Soa2Sore* [13], H

is also an SOA if we respect the symbols $\%_i^+$, $||_{ij}$ and $+_k$ ($i \in \mathbb{D}_\Sigma$, $j \in \mathbb{P}_\Sigma$ and $k \in \mathbb{B}_\Sigma$) as alphabet symbols let $S_\%^\#$ denote the language recognized by H. There is $\mathcal{L}(r_s) \supseteq \mathcal{L}(H) \supseteq S_\%^\#$ [13].

For a string $s \in S_\%^\#$, if the symbols $\%_i^+$, $||_{ij}$ and $+_k$ are removed from s to obtain a new string s', there is $s' \in \mathcal{L}(G)$ where G is the SOA that is built for S and then to be converted to the FAUC $\mathcal{A}$ ($\mathcal{L}(\mathcal{A}) \supseteq \mathcal{L}(G)$) in algorithm *ConsFauc*. There is $s' \in \mathcal{L}(\mathcal{A})$.

In algorithm *GenSoreuc*, the symbols $\%_i^+$, $||_{ij}$ and $+_k$ are removed from r_s step by step, and unorder concatenations and counting operators are introduced into r_s, let r denote the finally updated r_s. Every symbol in r occurs at most once, r is a SOREUC and there is $s' \in \mathcal{L}(r)$. Then, there is $\mathcal{L}(r) \supseteq \mathcal{L}(\mathcal{A})$. Since $\mathcal{L}(\mathcal{A}) \supseteq S$, there is $\mathcal{L}(r) \supseteq S$.