# A THE DETAILS OF PROOFS

## A.1 Proof of Proposition 1

*Proof.* According to Proposition 18 in [25], if $\Sigma$ is a finite alphabet of $n$ alphabet symbols, the number of pairwise non-equivalent SOREs over $\Sigma$ is $s(n)$ with $n!2^{3n-r\log n} \leq s(n) \leq n!2^{7n}$, where $r$ is a constant. This implies there is a finite number of non-equivalent SOREs.

For $k$-OREs, every symbol in $\Sigma$ occurs at most $k$ times, we treat the same symbol in a $k$-ORE as distinct. Then, let $\Sigma(k)$ for $k$-OREs be a finite alphabet of $nk$ alphabet symbols, the number of pairwise non-equivalent $k$-OREs over $\Sigma(k)$ is $s(nk)$ with $(nk)!2^{3nk-r\log(nk)} \leq s(nk) \leq (nk)!2^{7nk}$. This implies there is a finite number of non-equivalent $k$-OREs over $\Sigma(k)$. According to the definition 1, $k$-REs ($k \geq n$) is a subclass of $k$-OREs. the number of non-equivalent $k$-REs over $\Sigma(k)$ is also finite.

Let $\mathcal{D}$ denote the class of $k$-REs. Assume that there is a language $L \subseteq \Sigma_k^*$ such that no expression $\alpha \in \mathcal{D}$ is a descriptive generalization of $L$ (w.r.t. the class $\mathcal{D}$). If an expression $\alpha_1 \in \mathcal{D} : \mathcal{L}(\alpha_1) \supseteq L$, then there is an expression $\alpha_2 \in \mathcal{D} : \mathcal{L}(\alpha_1) \supset \mathcal{L}(\alpha_2) \supseteq L$. There are infinite expressions $\alpha_1, \alpha_2, \cdots, \alpha_i, \cdots \in \mathcal{D}$ such that $\mathcal{L}(\alpha_1) \supset \mathcal{L}(\alpha_2) \supset \cdots \supset \mathcal{L}(\alpha_i) \supset \cdots \supseteq L$. This contradicts the fact that there is only a finite number of non-equivalent $k$-REs over $\Sigma(k)$. Hence, for every language $L$, there exists a $k$-RE $r$ that is a descriptive generalization of $L$ (w.r.t. the class of $k$-REs).

## A.2 Proof of Theorem 1

*Proof.* In algorithm 1, a GA $\mathcal{A}$ is obained by deleting nodes and the corresponsding edges in the given directed graph $G$. The rule $R_1$ working on $G$ corresponds the operations in lines 13∼14. The rules $R_2$ and $R_3$ work on $G$ in line 21 and line 11, repectively. This implies that the finally obtained GA $\mathcal{A}$ is reducible. SCCs are searched in $G$ in a recusive way in line 4, and the characters about stable and transverse are added into $\mathcal{A}$ for each SCC in line 7. Actually, the algorithm $Trans2GA$ is a variant of algorithm $Soa2Sore$ [25]. Then, according to Theorem 27 in [25], for any given directed graph $G$, $G$ can be correctly transformed into an expression. In this paper, each unit able to be transformed into an expression has been processed by $Trans(G, \mathcal{A})$ until $|G.V| = 2$. The finally obtained GA $\mathcal{A}$ has the glushkov characters: HM, SS, ST and RD. According to Theorem 5.1 in [19], a dircted graph is a glushkov graph if and only if the directed graph satisfies characteristics of glushkov graph (glushkov characters) [19]. Hence, $\mathcal{A}$ is a glushkov graph.

## A.3 Proof of Theorem 2

*Proof.* For any given finite sample $S$ and the directed graph $G$ obtained by using $PSO$ algorithm, there does not exits another directed graph $G'$ such that $\mathcal{L}(G) \supset \mathcal{L}(G') \supseteq S$; otherwise, $G$ is not the optimal result of $PSO$ algorithm.

The directed graph $G$ obtained by using *PSO* algorithm is then transformed into a GA $\mathcal{A}$. Let $f$ denote the function used to transform $G$ into $\mathcal{A}$ (i.e., $f(G) = \mathcal{A}$). $\mathcal{A}$ must satisfy glushkov characters and any character is unambiguous. $f$ is unique, there does not exist another function $f'$ such that $f'(G) = \mathcal{A}$. It is implies that there does not exist another GA $\mathcal{A}'$ such that $G$ can be transformed into $\mathcal{A}'$.

For any given finite sample $S$, $\mathcal{L}(\mathcal{A}) \supseteq \mathcal{L}(G) \supseteq S$. There does not exist another GA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) \supset \mathcal{L}(\mathcal{A}') \supseteq S$. The constructed GA $\mathcal{A}$ is a descriptive generalization of $S$.

### A.4 Proof of Theorem 3

*Proof.* According to Theorem 2, for any given finite sample $S$, the obtained GA $\mathcal{A}$ is a descriptive generalization of $S$. $\mathcal{A}$ is a precise representation for $S$. In algorithm *LearnRE*, the algorithm *Soa*2*Sore* is used to transform GA $\mathcal{A}$ into an expression. For any given finite sample $S'$, since *Soa*2*Sore* can transform a directed graph $G$ which is a descriptive generalization of $S'$ into an expression that is also a descriptive generalization of $S'$ (Theorem 27 in [25]), *Soa*2*Sore* can also transform GA $\mathcal{A}$ into a $k$-RE that is also a descriptive generalization of $S$ (w.r.t. the class of $k$-REs). Hence, the $k$-RE $r_k$ returned by *LearnRE* is a descriptive generalization of $S$ (w.r.t. the class of $k$-REs).

### A.5 Proof of Corollary 1

*Proof.* For any $k$-RE $r_k$, we can construct an equivalent GA $A_k$, i.e., $\mathcal{L}(A_k) = \mathcal{L}(r_k)$. The GA $A_k$ is also a directed graph, we can obtain a finite sample $S$ by traversing $A_k$. For a string $s \in S$, $s$ is a path from $A_k$. Then, there exist a finite sample $S$ that can be obtained from $A_k$ such that $A_k$ is a descriptive generalization of $S$. When $S$ is as input of algorithm *LearnRE*, the obtained GA $\mathcal{A}$ is also a descriptive generalization of $S$ according to Theorem 2. There is $\mathcal{L}(\mathcal{A}) = \mathcal{L}(A_k) = \mathcal{L}(r_k)$. $\mathcal{A}$ equivalent to $r_k$ is then transformed into an expression $r$ in *LearnRE*, $r$ is also a descriptive generalization of $S$, it implies that $\mathcal{L}(r) = \mathcal{L}(A) \supseteq S$. Hence, there exists a expression $r$ can be generated by *LearnRE* such that $\mathcal{L}(r_k) = \mathcal{L}(r)$. $r_k$ can be successfully derived by *LearnRE*.