A GNFA, $(Q, \Sigma, S, q_{\text{start}}, q_{\text{accept}})$ is a 5-tuple s.t.

1. $Q$ is the finite set of states
2. $\Sigma$ is the input alphabet
3. $S : (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \rightarrow R$,
   $R$ is the set of all regular expressions over $\Sigma$
4. $q_{\text{start}}$ is the start state
5. $q_{\text{accept}}$ is the accept state
Lem. If a language is regular, then it is described by a regular expression.

IDEA: (1) DFA $\rightarrow$ GNFA
* (2) GNFA $\rightarrow$ regular expression

Sketch of proof (step 2):
"By construction",

Let $M$ be the DFA for language $A$, we first convert $M$ to a GNFA $G$.
Then, run $\text{Convert}(G)$:

1. Let $k$ be the # of states in $G$
2. If $k = 2$, return the expression $R$
3. If $k > 2$, select any state $q_i$, $p \in Q$
different from $q_\text{start}$ and $q_\text{accept}$.
   Delete $q_i$ as in Fig 1 to obtain a new GNFA $G'$ (with $k-1$ states)
4. Recursively call $\text{Convert}(G')$.