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

1. Q is the finite set of states
2. Σ is the input alphabet
3. $\delta: (Q - \{q_{accept}\}) \times (Q \to \{q_{start}\}) \to R$.
   R is the set of all regular expressions over Σ
4. q_{start} is the start state
5. q_{accept} is the accept state.

Lem. If a language is regular, then it is described by a regular expression.

IDEA: (1) DFA $\longrightarrow$ GNFA

* (2) GNFA $\longrightarrow$ regular expression

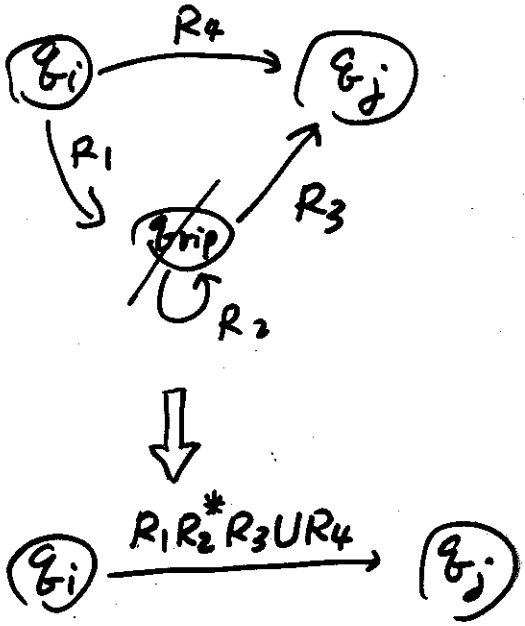

Fig 1

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 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 $q_{rip} \in Q$ different from q_{start} and q_{accept}.
   Delete q_{rip} as in Fig 1 to obtain a new GNFA G' (with $k-1$ states).
4. Recursively call Convert(G').