8. For languages $A, B \subseteq \Sigma^*$, define $\text{SHUFFLE}(A, B)$ to be the set

$$\{w \mid w = a_1 b_1 a_2 b_2 \cdots a_k b_k \text{ where } a_1 \cdots a_k \in A \text{ and } b_1 \cdots b_k \in B, \text{ with each } a_i, b_i \in \Sigma^*\}.$$ 

Show that the regular languages are closed under shuffle. Give both a short, convincing, one paragraph "proof idea" similar to those in the text, and a formal proof. Hint: A variant of the "Cartesian product" construction in Theorem 1.25 may be useful. And, yes, "induction is your friend."

Note: Read the definition carefully. It says "$a_1 \cdots a_k \in A$," not "$a_1, \ldots, a_k \in A$"; the later specifies $k$ strings, each individually in $A$; the former specifies $k$ strings, perhaps none in $A$, whose concatenation (in order) is a single string in $A$.

This works in part since, without loss of generality, every $(a_i, b_i)$ pair has $|a_i b_i| = 1$, i.e. one is epsilon, the other a single character.
Relating edges of $G'$ to paths in $G$

A path in $G$: any sequence of states
A simple path in $G$: any sequence of
  distinct states at 1st and last are not $k$,
  and all intermediate ones (if any) are $k$.

\[
\begin{align*}
  i &
  \rightarrow
  j \\
  i &
  \rightarrow
  k
  \rightarrow
  j \\
  i &
  \rightarrow
  k
  \rightarrow
  j
  \rightarrow
  j
\end{align*}
\]

The Point:

(a) every path in $G$ can be decomposed into simple paths

(b) every edge in $G'$, say $i \rightarrow j$,
  corresponds to the set of all
  simple paths in $G$ with those
  end points.
Q: What strings accepted by \( L_0 \rightarrow L_1 \rightarrow L_5 \)?

\[
\{ w \mid w = x_1 x_2 \text{ with } x_1 \in L_1 \land x_2 \in L_5 \}
\]

\( L_1 \circ L_5 \)

\( L_0 \rightarrow L_5 \rightarrow L_2 \rightarrow L_7 \# \rightarrow L_2 \)

\( L_1 \circ L_2 \circ L_3 \circ L_4 \circ L_5 \)

\( L = \bigcup_{p \text{ path of } L_0} \text{concat } \text{of } L_0 \text{'s on path } p \)
Claim 2

\[ L(r_{ij}) = \{ w \mid G \text{ can move from } i \text{ to } j \text{ reading } w \text{ and passing through no intermediate states except possibly } k \} \]

Equivalently:

\[ L(r'_{ij}) = \{ w \mid G \text{ can move from } i \text{ to } j \text{ reading } w \text{ along a simple path } \}

\[ = L(r_{ij} \circ r_{ik} \cdot r_{kk} \cdot r_{kj}) \]
Claim 2
\[ L \left( r_i u r_k r_k^* r_k^* i_j \right) \]
\[ = \{ x \mid \text{G could go from } i \text{ to } j \text{ without passing through any intermediate state except possibly } k \} \]

Claim 3
\[ G \& G' \text{ are equivalent} \]

Claim 4
\[ \exists \text{ an NFA equivalent regular expression.} \]

Proof: NFA → GNFA → 2-state GNFA → E.RE.
by induction on \( k \), using Claim 1
In this example we begin with a three-state DFA. The steps in the conversion are shown in the following figure.

(a) 

(b) 

(c) 

(d) 

(e) 

\[(a(aa \cup b)^*ab \cup bb)((ba \cup a)(aa \cup b)^*ab \cup bb)^*((ba \cup a)(aa \cup b)^* \cup \varepsilon) \cup a(aa \cup b)^*\]

**Figure 1.69**
Converting a three-state DFA to an equivalent regular expression
Summary

L is regular $\iff$

$L = L(M)$ for some DFA $M$

$L = L(N)$ $\iff$ NFA $N$

$L = L(G)$ $\iff$ GNEA $G$

$L = L(R)$ $\iff$ Regular $R$