- ($\Rightarrow$) For all regular languages, there exists a regular grammar:
  Suppose we have a regular language. Then that language is accepted by a DFA or accepted by an NFA or generated by a regular expression.
  - Construct a regular grammar (directly from the DFA/NFA, or by induction on the structure of the regular expression)
  - Prove that the constructed grammar and the given DFA/NFA/regular expression are equivalent.
    - Suppose w is accepted by the DFA/NFA/regular expression, show w is generated by the grammar.
    - Suppose w is generated by the grammar, show w is accepted by the DFA/NFA/regular expression.

    (If you suppose you have a regular expression, then the proof of equivalence can be interwoven with the construction of the grammar, inductively in the structure of the regular expression.)
- ($\Leftarrow$) For all regular grammars, the language generated is regular.
- Suppose we have a regular grammar.
  - Construct an NFA/DFA/regular expression
  - Prove the grammar and NFA are equivalent
    - Suppose w is generated by the grammar, show w is accepted by the DFA/NFA/regular expression.
    - Suppose w is accepted by the DFA/NFA/regular expression, show w is generated by the grammar.

Proof. *Suppose we have a regular language generated by regular expression R. We shall construct a regular grammar for it by induction on the structure of R.*

Case 1: *Suppose R is a single character a. Then $L(R) = \{a\}$, and the grammar $S \to a$ generates $L(R)$.*

Case 2: *Suppose R is the concatenation of two regular expressions R_1 and R_2. By structural induction, R_1 and R_2 can be generated by two regular grammars with start symbols S_1 and S_2. Then consider the grammar that is just like the union of these two grammars except for the rules of the first grammar that do not end in a non-terminal. For each such rule ($T_1 \to w_1$) and for each rule in the second grammar in the form $S_2 \to w_2 T_2$ and $S \to w_3$, make the new rule $T_1 \to w_1 w_2 T_2$ and $T_1 \to w_1 w_3$, respectively. Note that this grammar is regular. Moreover, since the two regular grammars generate the same languages as R_1 and R_2 by induction, then this new grammar also generates the same language as $R = R_1 R_2$.*

Case 3: *Suppose R is the union of two regular expressions R_1 and R_2. By structural induction, R_1 and R_2 can be generated by two regular grammars with start symbols S_1 and S_2. Then consider the grammar that is the union of these two grammars, but with new non-terminal and start symbol S and rules $S \rightarrow S_1$ and $S \rightarrow S_2$. Note this grammar is regular and generates $L(R)$.*
Case 4: *Suppose R is the Kleene closure of of some regular expression R_1. By structural induction there is a regular grammar (with start symbol S_1 that generates $L(R_1)$). Then construct the grammar just like grammar for R_1 except that it also has the rule $S_1 \rightarrow e$ (if it doesn't already) and that for every rule in the form $T \rightarrow w$, change that rule to be $T \rightarrow wS_1$. Note that this grammar is regular and, by how it is constructed, generates the same language as $R = R_1*$.*

Conversely, suppose we have a regular grammar. Then construct a non-deterministic finite automaton that has a state for every nonterminal in the grammar.
For every rule in the grammar in the form $T_1 \rightarrow T_2$*, add a transition from* T_1 *to* T_2 *with e. For every rule in the form* $T_1 \rightarrow a_1 \ldots a_n$*, add states labelled "*a_1, a_2*" ... "*a_{n-1}, a_n*" and transitions from* T_1 *to* a_1, a_2 *with* a_1*, and for all* i*,* $1 \leq i < n-1$*, transitions from* a_i, a_{i+1} *to* a_{i+1}, a_{i+2} *with* a_{i+1}*; also, make states* a_{n-1}, a_n *accept states. Finally, for every rule in the form* $T_1 \rightarrow a_1 \ldots a_n T_2$*, do as in the previous case but instead of making* a_n *an accept state, add a transition from* a_{n-1}, a_n *to* T_2 *with* a_n*.*

Now to prove that this constructed NFA is equivalent to the grammar. First suppose w is a string in the language generated by the grammar. That means there exists a parse tree consistent with the grammar that yields w. Consider any internal node in the tree, say labelled with nonterminal A, which corresponds to a state in our construction. There must be a rule in the grammar corresponding to the expansion at this node, either in the form $A \rightarrow abc$ or $A \rightarrow abcB$. If the former, then in our construction there is a path taking those terminals to an accept state. If the latter, then in our construction there is a path taking those terminals to a state B. Applying this observation inductively, our constructed NFA accepts w.

On the other hand, suppose our constructed NFA accepts string w. We can take the route of the string through the NFA and reconstruct a parse tree: every sequence of transitions starting from a non-terminal state, ending at a non-terminal state, and having no non-terminal states in between corresponds to a node in the (alleged) parse tree. Each of these comes from a rule in the grammar. $\square$