Concepts of Programming Languages, CSCI 305, Fall 2021
Translation of Regular Expressions to a DFA, Sept. 8
Section 2.2.1 Generating a Finite Automaton, 56-61

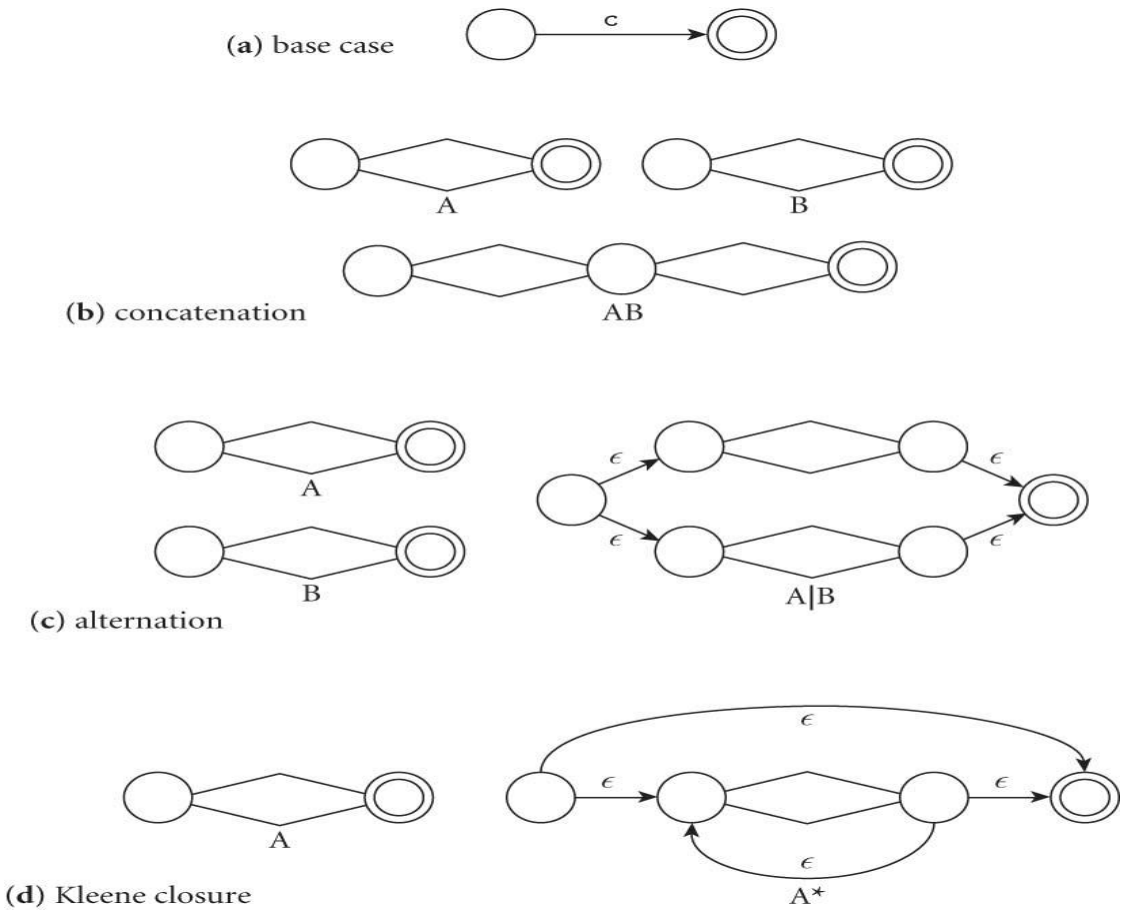
Finite Automaton – "machine" representing regular expressions

- Consists of states and arrows between states (transitions)
- One start state
- Any number of accept states (use double circle to indicate the state is accepting)
- Automaton is used to show acceptance and non-acceptance of strings

Deterministic Finite Automaton, DFA – Given any state and symbol, there is exactly one transition

Non-deterministic Finite Automaton, NFA – Given a state and symbol there may be many or no transitions. Also, ε -transitions are allowed (transitions that can be taken without consuming a symbol). Zero or more ε –transitions can be applied before or after a transition.

Every regular expression can be translated to an NFA



Every NFA can be translated to a DFA

Algorithm to go from an NFA to a DFA:
In general make the states of the DFA represent the set of states that the NFA could be possibly be in.

1. Create an initial state which consists of the initial state of the NFA along with all of the states that the NFA could have been in initially (i.e. follow epsilon transitions if possible)
2. Repeat until transitions have been defined for every state in the DFA on every input.

   For each state q (represented by a set of states) and each input a, create a transition from q on a which goes to a state representing all states that the NFA could have been in if it started in a state in q and saw the input a. Be sure to follow epsilon transitions as well.
3. Mark any state whose representation contains a final state in the NFA as final.