2.4. NONDETERMINISTIC FINITE AUTOMATA

In the start state, on input character a, the automaton can move either right or left. If left is chosen, then strings of a’s whose length is a multiple of three will be accepted. If right is chosen, then even-length strings will be accepted. Thus, the language recognized by this NFA is the set of all strings of a’s whose length is a multiple of two or three.

On the first transition, this machine must choose which way to go. It is required to accept the string if there is any choice of paths that will lead to acceptance. Thus, it must “guess,” and must always guess correctly.

Edges labeled with $\epsilon$ may be taken without using up a symbol from the input. Here is another NFA that accepts the same language:

![Diagram of another NFA accepting the same language]

Again, the machine must choose which $\epsilon$-edge to take. If there is a state with some $\epsilon$-edges and some edges labeled by symbols, the machine can choose to eat an input symbol (and follow the corresponding symbol-labeled edge), or to follow an $\epsilon$-edge instead.

CONVERTING A REGULAR EXPRESSION TO AN NFA

Nondeterministic automata are a useful notion because it is easy to convert a (static, declarative) regular expression to a (simulatable, quasi-executable) NFA.

The conversion algorithm turns each regular expression into an NFA with a tail (start edge) and a head (ending state). For example, the single-symbol regular expression $a$ converts to the NFA

![Diagram of NFA for a]

The regular expression $ab$, made by combining $a$ with $b$ using concatenation, is made by combining the two NFAs, hooking the head of $a$ to the tail of $b$. The resulting machine has a tail labeled by $a$ and a head into which the $b$ edge flows.