Formalizing $\varepsilon$-NFA

**Defn:** An $\varepsilon$-NFA $N$ consists of:

- Same as a DFA: A set $Q$ of states, an alphabet $\Sigma$, a start state $q_0 \in Q$, a set of accept states $A \subseteq Q$
- A transition function $\delta : Q \times \Sigma \to 2^Q$
- An $\varepsilon$-transition function $\varepsilon : Q \to 2^Q$

Given an $\varepsilon$-NFA, we define:

- $\bar{\varepsilon} : 2^Q \to 2^Q$, where $\bar{\varepsilon}(S)$ gives the set of states reachable from $S$ using only $\varepsilon$-transitions (the $\varepsilon$-closure of $S$)
- $\hat{\delta} : Q \times \Sigma^* \to 2^Q$, where $\hat{\delta}(q, x)$ gives the set of states reachable from $q$ by processing $x$
  - $\hat{\delta}(q, \varepsilon) := \bar{\varepsilon}(q)$, and $\hat{\delta}(q, xa) := \bar{\varepsilon} \left( \bigcup_{q' \in \hat{\delta}(q, x)} \hat{\delta}(q', a) \right)$.
- $M$ accepts $x$ if there is an accept state in $\hat{\delta}(q_0, x)$.
Removing nondeterminism

Claim: if \( L \) is \( \varepsilon \)-NFA recognizable, then \( L \) is DFA-recognizable

Proof idea: Given an NFA \( N = (Q, \Sigma, \delta_N, q_0, F_N, A_N) \), build a DFA \( M \) with the same language. NFA can be in a set of states, while a DFA can be in only one state, so we make each single state of \( M \) represent a set of states of \( N \).

\[
Q = \mathcal{P}(Q_N) \\
\Sigma = \{a, b, c\} \\
\delta_M(q_0, \varepsilon) = q_{0M} \subseteq Q \\
\delta_M((q, a), b) = \bigcup_{q \in q} \delta_N(q, b) \\
\delta_M(q, \varepsilon) = q_{0M}
\]

\( A_M = \{q | q \text{ contains an elt. of } A_N \} \)

Claim: \( L(M) = L(N) \)

Subclaim: if \( \delta_M(q_{0M}, \varepsilon) = S \) then \( \delta_N(q_{0M}, \varepsilon) = S \)

More concisely: \( \delta_M(q_{0M}, \varepsilon) = \delta_N(q_{0M}, \varepsilon) \)

Proof: structural induction.

Proof of claim from subclaim:

plug in def. for \( L(M), L(N), \) and subclaim, \( A_M \).
Regular expressions

License plate: 3 letters followed by 4 digits
\[ [a-zA-Z] \]
\[ [0-9] \]
\[ [a-zA-Z][a-zA-Z][0-9][0-9][0-9][0-9] \]

Number \[ [0-9]^* \] 0 or more repetitions.

Month: "Jan" or "Feb" or "Mar" or...
Formal definition of a regular expression / language of a regular expression

$$r \in \text{RE} := \emptyset | \varepsilon | a | r_1 r_2 | r_1 + r_2 | r^*$$

$$a \in \Sigma$$

"Ap" = (a, r) r

any number of

strings, each matching

month: Jan + Feb + Mar + ...

digit: 0+1+2+...+9

number: (digit)* \(\Rightarrow\) matches \(\varepsilon\)

number': (digit)(digit)*

number'' : (1, 2, ..., 9)(digit)*

Let \(L : \text{RE} \rightarrow 2^\Sigma^*\) be given by: \(\text{cat}(xy)\) can use

\(L(\emptyset) := \emptyset\)

\(L(\varepsilon) := \{\varepsilon\}\)

\(L(a) := \{\varepsilon a \} \quad \text{or} \quad \{a\}\)

\(L(r_1 r_2) := \{xy \mid x \in L(r_1) \text{ and } y \in L(r_2)\}\)

\(L(r_1 + r_2) := L(r_1) \cup L(r_2)\)

\(L(r^*) := \{\varepsilon\} \cup \{x_1 x_2 x_3 \ldots x_n \mid \text{ each } x_i \in L(r)\}\)

\(n \geq 0 \) repetitions of \(r\).

\(L(r)\) is the language of \(r\). We say

\(x\) matches \(r\) or \(r\) matches \(x\) if \(x \in L(r)\).
\[ L = \{ x \mid x \text{ has an odd } 1 \text{'s} \} \]

give a RE for \( L \).

\[ \overline{0^*10^* (0^*10^*)^*} \]