Introduction to finite-state acceptors for strings
(the non-recursive way)

Non-deterministic finite-state string acceptors

Definition 1. *A* non-deterministic finite-state acceptor (NFA) *is a tuple* $(Q, \Sigma, I, F, \delta)$ *where*

- Q *is a finite set of states;*
- Σ *is a finite set of symbols (*the alphabet*);*
- $I \subseteq Q$ *is a set of* initial (start) *states;*
- $F \subseteq Q$ *is a set of* accepting (final) *states; and*
- $\delta \subseteq Q \times \Sigma \cup \{\epsilon\} \times Q$ *is the* transition relation.

A transition (q, a, r) means that if the acceptor is in state q and reads the symbol a in a string that it changes into state r, and advances to the next symbol in the string. A transition (q, ϵ, r) denotes a "free" change of state; that is the state of the system can change without any symbol in the input string being consumed. You can think of ϵ like the empty string. The basic idea is that a NFA accepts/recognizes/generates a string w if there is a sequence of transitions from a start state to a final state which consuming the w.

Formally, A NFA accepts/recognizes/generates a non-empty string w of length n if and only if there is a *path* – that is, a sequence of transitions – of length $m \geq n$

$$(q_0, x_1, q_1), (q_1, x_2, q_2) \ldots (q_{m-1}, x_m, q_m)$$

such that

- $q_0 \in I$,
- $q_m \in F$,
- for each $1 \leq i \leq m$ it is the case that $(q_{i-1}, x_i, q_i) \in \delta$
- and $x_1 x_2 \ldots x_m = w$

A NFA accepts/recognizes/generates the empty string if and only if $I \cap F \neq \varnothing$.

Exercise 1 Write NFA for the sets of strings in Example 1 #1-#13 in Chapter 2.

0.1 Deterministic finite-state string acceptors

A NFA is *deterministic* if and only if

- $|I| \leq 1$ (at most one start state),
- If $(q, x, r) \in \delta$ then $x \neq \epsilon$ (no epsilon transitions), and
- If $(q, a, r), (q, a, r') \in \delta$ then $r = r'$ (at most one a-transition for each state).

We write DFA for deterministic finite-state acceptors.

Exercise 2 Which of the NFA you provided in Exercise 1 are deterministic?