NFA, \(\varepsilon\)-NFA, Subset Construction, \(\varepsilon\) Closure

Winter 2019

1 NFA and \(\varepsilon\)-NFA

A Non-deterministic Finite Automaton (NFA) is a 5-tuple \((\Sigma, Q, q_0, A, \delta)\) where:

- \(\Sigma\) - a non-empty finite set of symbols, the input alphabet
- \(Q\) - a non-empty finite set of states.
- \(q_0 \in Q\) - the starting state
- \(A \subseteq Q\) - set of final states.
- \(\delta: Q \times \Sigma \rightarrow P(Q)\) - The transition function (some use \(2^Q\) to represent the power set of \(Q\) rather than \(P(Q)\)).

The key difference between NFAs and DFAs is the transition relation. For DFA, the transition relation is \(\delta: Q \times \Sigma \rightarrow Q\), each \(\delta(q, \sigma)\) gives a single state for any \(q \in Q\) and \(\sigma \in \Sigma\).

For NFA, each \(\delta(q, \sigma)\) is a set of states, i.e., an element in the power set \(P(Q)\). Recall what power sets are: \(P(\{x, y, z\}) = \{\{\}, \{x\}, \{y\}, \{z\}, \{x, y\}, \{x, z\}, \{y, z\}, \{x, y, z\}\}\). This means that given some input, an NFA can be in multiple states at once.

1.1 \(\varepsilon\)-NFA

An \(\varepsilon\)-NFA allows for the use of \(\varepsilon\)-transitions. \(\varepsilon\)-transitions represent the transition from one state to another without consuming any input. This is useful when we want to try to connect multiple NFAs into one.

\(\varepsilon\)-NFAs have a very similar definition as NFAs, but we add \(\varepsilon\)-transitions for every state to the transition relation: \(\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)\)

1.1.1 Exercise

1. Show that every DFA is also an NFA.

2. Draw NFAs for the following languages over \(\Sigma = \{0, 1\}\)
   (a) Let \(A\) be the language of strings ending in 1011.
   (b) Let \(B\) be the language of strings ending in either 10 or 01.
   (c) Let \(C\) be the language of strings beginning with 1000.

3. Draw an \(\varepsilon\)-NFA for the language \(C(A \cup B)\), where \(A\), \(B\) and \(C\) are languages from the previous exercise.
2 NFA to DFA conversion

Recall the definition of a DFA: $(\Sigma, Q, q_0, A, \delta)$ where:

- $\Sigma$ - a non-empty finite state of symbols, the input alphabet
- $Q$ - a non-empty finite set of states.
- $q_0 \in Q$ - the starting state
- $A \subseteq Q$ - set of final states.
- $\delta : Q \times \Sigma \rightarrow Q$ - The transition function.

Furthermore, recall the only difference for an NFA is that $\delta$ outputs a set of states (that is, $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ instead of a single state. This means that we can reinterpret an NFA as a DFA where the states in the DFA are sets of states in the NFA. This is called the subset construction.

While in theory we could simply draw all $2^{|Q|}$ states in the DFA, this can be very time-consuming and most of the states will not be necessary in practice. Instead we use the following algorithm to convert the DFA $N = (Q', \Sigma, q'_0, A', \delta')$ from NFA $M = (Q, \Sigma, q_0, A, \delta)$:

1. Create the start state $q'$ of the DFA and label it $\{q'\}$, where $q_0$ is the starting state of the NFA.
2. Let $S = \{q_0\}$
3. Add $S$ as a state of $Q'$
4. For each letter $a \in \Sigma$, let $T$ be the set of states reachable by consuming $a$ when starting at some state in $S$. Add a transition from the state $S$ to the state $T$, adding $T$ to the set of states if necessary. Mathematically:

$$\delta'(S, a) := \bigcup_{\sigma \in S} \delta(\sigma, A)$$

5. Repeat from step 3 with another state $S$ which has not yet ran by step 3, until no such state exist.
6. Label every state which contains an accepting states from $Q$ as accepting. Mathematically:

$$A' := \{S \in \mathcal{P}(Q) : S \cap A \neq \emptyset\}$$
2.1 Exercise

Convert the following NFA to DFA using the subset construction algorithm.

3 \(\varepsilon\)-NFA to NFA conversion

Similar to converting NFAs to DFAs, we also convert \(\varepsilon\)-NFA to NFA using \(\varepsilon\) closure algorithm to convert the NFA \(N = (Q', \Sigma, q'_0, A', \delta')\) from \(\varepsilon\)-NFA \(M = (Q, \Sigma, q_0, A, \delta)\):

1. Draw the start state of the NFA using the same start state as the \(\varepsilon\)-NFA.
2. Let \(Q'\) be the start state \(q_0\) as well as any other state \(q \in Q\) which has at least one non-\(\varepsilon\) transitions entering it.
3. From each state in \(q \in Q'\) and every letter \(a \in \Sigma\), add a transition to all states which can be reached from \(q\) by consuming exactly one \(a\).
4. Mark each state \(q \in Q'\) as accepting if it can reach an accepting state in the original \(\varepsilon\)-NFA by only following \(\varepsilon\) transitions.

3.1 Exercise

Convert the following \(\varepsilon\)-NFA to an NFA using the \(\varepsilon\) closure algorithm.