Chapter 8

NFAs With ϵ-transitions

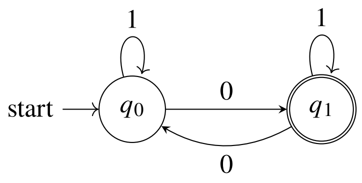
The last generalisation of a DFA that we are going to consider are NFAs with ϵ-transitions. Compared to the DFAs, NFAs had a feature that they can be in multiple states at the same time, and that some transitions can lead to an empty set of states (in effect, some transitions might be missing, so there does not have to be a transition out of every state for every input symbol). But still, all the transitions in an NFA have to happen on reading symbols. This means that an NFA makes a transition only as a response to reading an input symbol. This makes it a bit cumbersome to design NFAs to accept some languages. Consider the following example.

Example 8.1: [Odd 0's and 1's divisible by 3

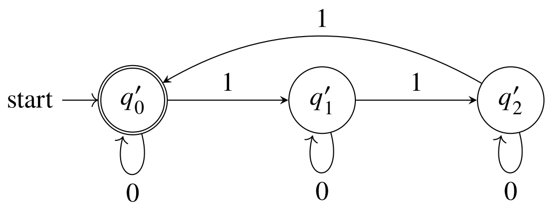
Design an NFA that accepts the language L of all strings over the alphabet $\{0, 1\}$ that have the odd number of 0's or the number of 1's divisible by 3.

$$L = \{w | w \in \{0, 1\}^*, w \text{ has } 2k + 1 \text{ 0's or } 3m \text{ 1's for some } k \text{ and } m\}.$$

Examples of strings from L are ϵ, 0, 111, 0101101, 1101110001, 1100101, and the examples of strings that are not il L are 001, 0011, 110110. It is reasonalby easy to find an NFA that accepts L. We can have six states in it, which correspond to the number of 0's seen so far moduo 2, and the number of 1's seen so far moduo 3. For example, the state $q_{0,2}$ would correspond to the even number of 0's and the number of 1's which gives remainder 2 when divided by 3 (so, for example, strings 0011, 11111, 000011, 1101011 etc.). The states would then be $q_{0,0}, q_{0,1}, q_{0,2}, q_{1,0}, q_{1,1}, q_{1,2}$, with $q_{0,0}$ being both the start state and the sole accepting state. It is also quite easy to derive the transitions for this NFA. However, it feels like this should be easier to do. Our language L is really the union of two languages - language L_1 of all strings that have the odd number of 0's and the language L_2 of all strings that have the number of 1's divisible by 3. It is pretty trivial to come up with NFAs for L_1 and L_2. One NFA that accepts L_1 is given by

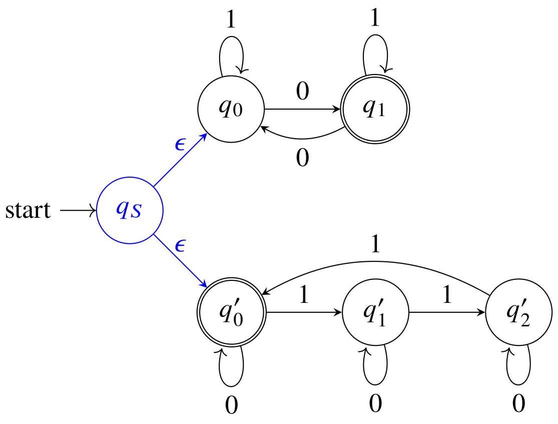


One NFA that accepts L_2 is given by



It would be ideal if we could just join these two NFAs together into a new NFA that would have each of these NFAs as one path in it, and would accept the string if that string is accepted by either of these two paths.

However, that is not trivial to do. One of the problems is that our NFA has to have a single start state. So, we would need to insert a new start state and then somehow, based on the first symbol read, split the new NFA into two paths. Much easier solution would be if we had a new start state, then from this state spontaneously transit (that is, transit without reading any symbol of our input string) to the starting states of both the NFA for L_1 and the NFA for L_2 and finally run both of the NFAs in parallel. Our automaton would look like this



Here, q_S is the new start state and the transitions labelled with ϵ are spontaneous transitions that are made without reading any input symbol. Here, we have a single start state q_S and from that state we immediately transit to start states of the two NFAs. Then, as we read symbols of our input string, we would essentially run these two NFAs in parallel, making transitions that each of them would do. NFAs with ϵ-transitions allow us to do exactly this.

8.1 Formal Definition of a NFA With ϵ-transitions

An NFA with ϵ-transitions is formally defined as follows.

Definition 8.1: NFA With ϵ-transitions

Non-Deterministic Finite Automaton with ϵ-transitions is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of *states*
- Σ is a finite *alphabet* (set of possible input symbols)
- $\delta : Q \times \Sigma \cup \{\epsilon\} \to \mathcal{P}(Q)$ is the transition function
- $q_0 \in Q$ is the *start state*
- $F \subset Q$ is the set of *accepting states*

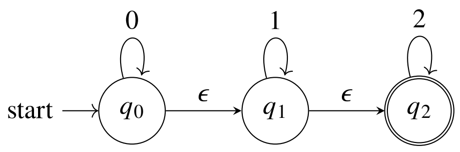
Compared to the definition of a basic NFA, the only difference is that the transition function δ is now defined for all symbols of the alphabet Σ *and* the empty string ϵ. That is, for each state $q \in Q$, we have transitions $\delta(q, a)$ for each $a \in \Sigma$ plus the transition on empty string, $\delta(q, \epsilon)$. We call this an ϵ-transition in q. Of course, as is the case with basic NFAs, $\delta(q, \epsilon)$ can also be $\emptyset$ (which is, in essence, absence of transition from q on ϵ).

How do the ϵ-transitions work? Whenever the NFA with ϵ-transitions enters a state q in which there is a non-trivial ϵ-transition (that is, the ϵ-transition that does not return $\emptyset$) to states $p_1, p_2, \dots, p_n$, we can see this as another source of determinism. The automaton stays in the state q, but also immediately transitions to all of the states $p_1, p_2, \dots, p_n$. Using our analogy that we had in basic NFAs, we can imagine this as creating multiple copies of our automaton, one which stays in the state q and one for each of the states $p_1, p_2, \dots, p_n$. All of these copies proceed in parallel. Of course, some of the states $p_1, p_2, \dots, p_n$ can have their own ϵ-transitions, so this process may continue, as we would also then immediately make all these transitions too.

Example 8.2: Example of an NFA With ϵ-transitions

Design an NFA with ϵ-transitions that accepts the language L over the alphabet $\Sigma = \{0, 1, 2\}$ of strings of form $0^n 1^m 2^l$, where $n, m, l \geq 0$.

The NFA with ϵ-transitions that accepts this language is given in the following transition diagram.



Operation of this automaton is shown in the following video.

Video 8.1: Example of an Operation of an NFA With ϵ-transitions

Example of an operation of the automaton in Example 8.2 on strings 001122 and 0011221.

8.2 Transition Function for an NFA With ϵ-transitions

As with the other kinds of finite automata, we also want to define a transition function for strings for NFAs with ϵ-transitions, so that we can formally describe the language accepted by such an NFA. Also, as usual, we want this transition function to return, for a given state q and a given string w, the set of states in which the automaton is if it starts in the state q and after it reads the string w. Our first intuition could be to define this transition function in exactly the same way as we did for the basic NFAs. That is, to have $\hat{\delta}(q, \epsilon) = \{q\}$ and $\hat{\delta}(q, xa) = \bigcup_{i=1}^{m} \delta(p_i, a)$ when $\hat{\delta}(q, x) = \{p_1, p_2, \ldots, p_m\}$. Alas, this does not work, as in this way we would ignore ϵ-transitions. In particular, the part $\hat{\delta}(q, xa) = \bigcup_{i=1}^{m} \delta(p_i, a)$ would give us a set of states that our NFA with ϵ-transitions reaches from the set of states $\hat{\delta}(q, x)$ on input symbol a, but not also all the states that it reaches from there following ϵ transitions. Thus, it does not give us all the states that the automaton is in after it ends reading the complete string xa. The same holds for the base case - $\hat{\delta}(q, \epsilon)$ would return just the state q, but not also all the states to which the automaton spontaneously transits from that state. So it would not accurately represent all the states in which the automaton is when it reads nothing.

To illustrate this, let us look at our example of a NFA with ϵ-transitions in Example 8.2. If $\hat{\delta}$ was defined in the above way, then $\hat{\delta}(q_0, \epsilon)$ would be just $\{q_0\}$. However, before it even reads anything, the automaton also transitions to the states q_1 and q_2, following the ϵ-transitions from q_0. Therefore, $\hat{\delta}(q_9, \epsilon)$ should really be $\{q_0, q_1, q_2\}$. Therefore, need a definition of a transition function that will include all the states reachable from a given set of states via ϵ transitions. In order to arrive to this definition, we first introduce a definition of an ϵ-closure. To define the ϵ-closure of a state precisely enough for formal reasoning, we need a recursive definition, but we also give a more intuitive, informal definition.

Definition 8.2: ϵ-closure of a State - Informal

ϵ-closure of a state q of an NFA with ϵ-transitions $M = (Q, \Sigma, \delta, q_o, F)$ is the set of all states from Q that are reachable from the state q using only ϵ transitions.

Definition 8.3: ϵ-closure of a State - Formal

Let q be a state of an NFA with ϵ-transitions $M = (Q, \Sigma, \delta, q_o, F)$. Then ϵ-closure of q can be defined in the following way:

(1) q is in ϵ-closure of q.

(2) If p is in ϵ-closure of q and $\delta(p, \epsilon) = r$ for some state r, then r is also in ϵ-closure of q.

(3) Nothing else is in ϵ-closure of q.

The definition of an ϵ-closure can intuitively be extend to sets of states.

Definition 8.4: ϵ-closure of a Set of States

ϵ-closure of the set of states $\{q_1, q_2, \ldots, q_n\}$ is the union of ϵ-closures of all the states in that set.

We denote ϵ-closure of q (or set of states P) by ϵ-CLOSURE(q) (or ϵ-CLOSURE(P)).
Let us calculate some ϵ-closures for the NFA with ϵ-transitions from Example 8.2:

$$\epsilon\text{-CLOSURE}(q_0) = \{q_0, q_1, q_2\}$$
$$\epsilon\text{-CLOSURE}(q_1) = \{q_1, q_2\}$$
$$\epsilon\text{-CLOSURE}(q_2) = \{q_2\}$$
$$\epsilon\text{-CLOSURE}(\{q_0, q_1\}) = \epsilon\text{-CLOSURE}(q_0) \cup \epsilon\text{-CLOSURE}(q_1) =$$
$$= \{q_0, q_1, q_2\} \cup \{q_1, q_2\} = \{q_0, q_1, q_2\}$$

ϵ-closure gives us a set of all states reachable from a given state, or a set of states, using only ϵ-transitions. This is exactly what was missing in our first attempt at a definition of a transition function for strings for NFAs with ϵ-transitions. Therefore, we are now ready to properly define a transition functions for strings for NFAs with ϵ-transitions.

Definition 8.5: Transition Function for Strings for an NFA With ϵ-transitions

Let $M = (Q, \Sigma, \delta, q_o, F)$ be an NFA with ϵ-transitions. We define the *transition function for strings over* Σ for this automaton, $\hat{\delta} : Q \times \Sigma^* \to \mathcal{P}(Q)$, in the following way:

- $\hat{\delta}(q, \epsilon) = \epsilon$-CLOSURE$(q)$.
- Let $\hat{\delta}(q, x) = \{p_1, p_2, \ldots, p_m\}$. Then

$$\hat{\delta}(q, xa) = \epsilon\text{-CLOSURE}(\bigcup_{i=1}^{m} \delta(p_i, a)).$$

To start explaining this definition, let us first consider in what ways does this transition function differ from a transition function of a basic NFA for strings. In the basic NFA, the transition function for a single state and the empty string ϵ always returns the set that contains only that single state. For the NFA with ϵ-transitions, the transition function for a single state and the empty string returns that same single state *plus* all the states reachable from it using only ϵ-transitions. For the string of the form xa, the transition function for basic NFAs returns the set of states reachable by following a transition on input symbol a from any state that is reached after reading the string x. The transition function for NFAs with ϵ-transitions returns this set, plus all the states reachable from the states of that set using ϵ-transitions. Therefore, in order to find $\hat{\delta}(q, xa)$ (the set of states in which the NFA with ϵ-transitions is after reading the complete string xa), we

- Find the set of states in which the NFA with ϵ-transitions is after reading the string x, $\{p_1, p_2, \ldots, p_m\}$.
- Follow the transitions from each of these states on input symbol a, giving us a new set of states $\{q_1, q_2, \ldots, q_n\}$.
- Find the ϵ-closure of each state q_i.
- Take the union of all of these ϵ-closures.

This is demonstrated in the following video.

Video 8.2: Example of Transition Function an NFA With ϵ-transitions

Example of applications of transition functions of the automaton in Example 8.2 on strings 001122 and 0011221.

8.3 String Acceptance and Language Accepted by a NFA With ϵ-transitions

Now that we have defined properly the transition function for strings for NFAs with ϵ transitions, we can define precisely the string acceptance by a NFA with ϵ-transitions, as well as the language accepted by a NFA with ϵ-transitions. Both definitions are pretty much the same as for the basic NFA.

Definition 8.6: String Acceptance by a NFA With ϵ-transitions

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA with ϵ-transitions. String $w \in \Sigma^*$ is accepted by M if $\hat{\delta}(q_0, w)$ contains at least one accepting state from F.
More formally, w is accepted by M if $\hat{\delta}(q_0, w) \cap F \neq \emptyset$.

Therefore, string w is accepted by a NFA with ϵ-transitions if there exists a sequence of transitions on the successive symbols of w (interspersed with ϵ transitions) from the starting state to one of the accepting states of the automaton.

Definition 8.7: Language Accepted a NFA With ϵ-transitions

et $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA with ϵ-transitions. Language L accepted by M, denoted by $L(M)$ or L_M, is the set of all strings accepted by M.

$$L(M) = \{w | w \in \Sigma^*, \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$