Chapter 3

Epsilon-NFAs

There is an extension that can be made to NFAs, that improves expressiveness, but does not add any fundamentally new capabilities. The idea of the extension is to allow an NFA to make a transition from one state to another *without* the need for an input symbol. We can think of this as *a transition caused by the empty string*, ϵ, which is why it is called an *ϵ-transition*.

An ϵ-NFA has the same capabilities as a standard NFA, because the class of languages that can be processed remains the same.

3.1 An informal view

We begin with an example: a signed integer is a string of characters. The first character can be an optional $+$ or $-$ sign, which is then followed by a sequence of decimal digits. Some examples of valid signed integers are: 12, -5, $+163$, and 9. Some illegal values are: $34A$, $-$, $-368-$, $3+$, and $3+4$.

Fig. 3.1 shows an ϵ-NFA, named $intRecog$, that can recognise a valid signed integer.

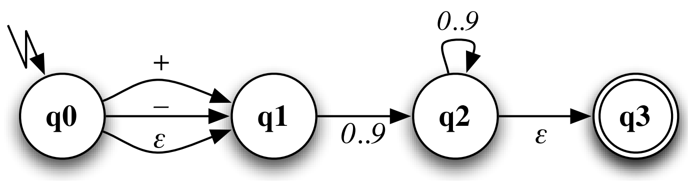


Figure 3.1: The intRecog ϵ-NFA

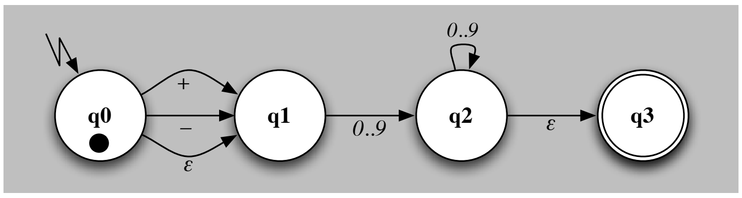
The diagram is very similar to the NFAs we studied previously, except it has two transitions labelled with ϵ. If the machine is in state q_0, it is permitted to make an ϵ-transition to state q_1, without requiring an input symbol. Similarly, if the machine is in state q_2, it can transition to state q_3 without requiring an input symbol.

The input alphabet for this NFA is $\Sigma = \{+, -, 0, 1, 2 \ldots 9\}$. (Notice that the alphabet does *not* include ϵ.) The set of states is $Q = \{q_0, q_1, q_2, q_3\}$. The initial state-set is $Q_0 = \{q_0\}$, and the set of accepting states is $F = \{q_3\}$

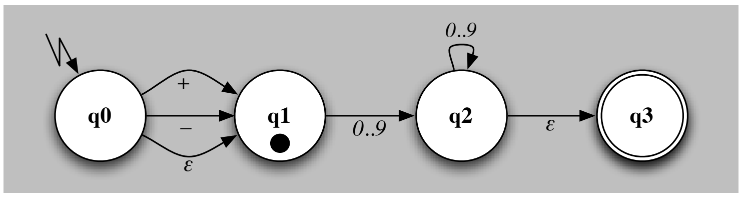
3.1.1 Example - the intRecog ϵ-NFA processing a string

Suppose we wish to recognise the string "+14".

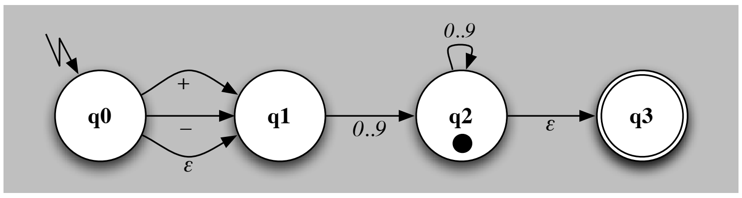
After initialisation (by the “lightnin strike”), the machine enters the state-set $\{q_0\}$. We can represent this with a dot on the diagram, like this:



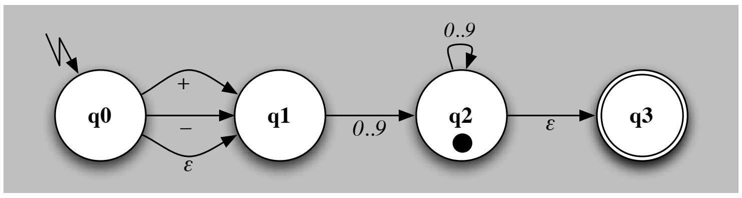
After processing the symbol $+$, the machine changes to the state-set $\{q_1\}$, and the diagram looks like this:



After processing the symbol 1, the machine changes to state-set $\{q_2\}$, and the diagram changes to this:



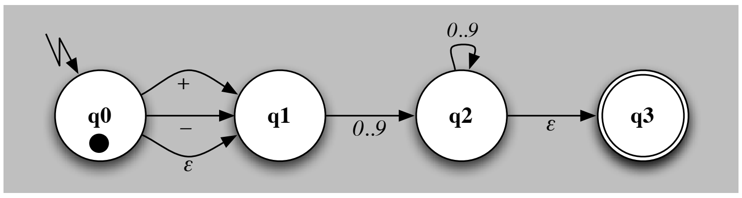
After processing the symbol 4, the machine remains in state-set $\{q_2\}$, and the machine looks like this:



If we ask “was this string recognised?”, it seems that the answer is “no”, becasue state q_2 is not an accepting state. However, there is an ϵ-transition from q_2 to q_3, so the machine *could* just transition to state q_3. Therefore we see that the machine *has* (correctly) recognised $+14$ as a valid string.

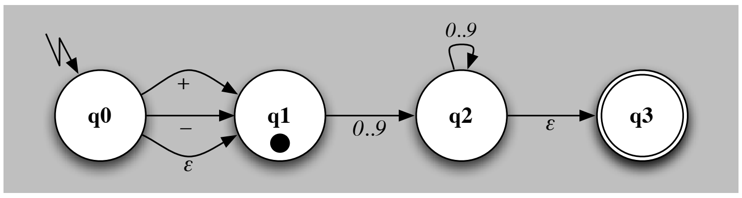
Now consider how it processes the string “65”.

Once again, after initialisation, the machine begins in the state-set $\{q_0\}$, like this:

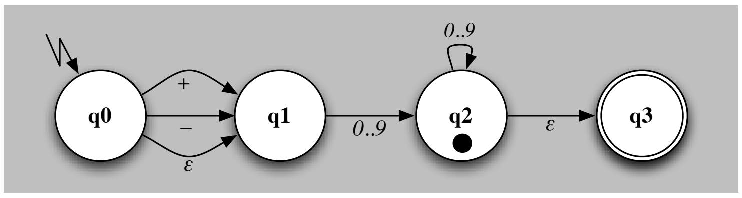


The machine now receives the symbol 6. Since there is no transition to handle this symbol from state q_0, a normal NFA would “die” at this point.

However, we have an ϵ-transition that we can take from state q_0 to q_1. After doing so, we arrive in state q_1.



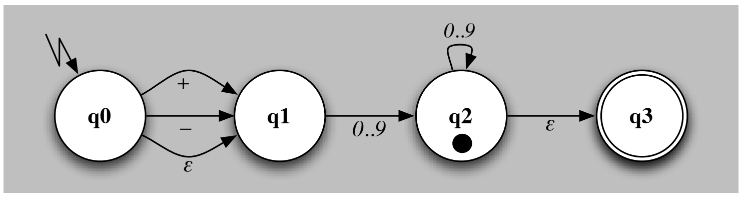
Now we are able to process the symbol 6, which takes us to state q_2, like this:



Notice that after taking an ϵ-transition, the rules require that the machine *must* process the symbol

6 by taking a normal transition. (If there had been a chain of epsilon transitions, the machine could take as many of them as it wanted. However, at the end it *must* process the 6.)

After processing the symbol 5, the machine remains in state-set $\{q_2\}$, and the machine looks like this:



Once again, because there is an ϵ-transition from q_2 to q_3, the machine has correctly recognised 65 as a valid string.

3.2 Formal definitions of an ϵ-NFA

3.2.1 5-tuple definition of an ϵ-NFA

An ϵ-NFA, E, can be defined by the tuple:

$$E = (Q, \Sigma, \delta, Q_0, F)$$

The meanings of Q, Σ, Q_0, and F are the same as for an NFA, but the definition of δ, is altered slightly: it accepts a state in Q, and *either* an input symbol s ($s \in \Sigma$) *or* ϵ, and returns a set of states. We can express this formally:

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \{Q\}$$

where $\{Q\}$ is a set of sets of states, i.e. the *set of all subsets* of Q.
For example, for the intRecog ϵ-NFA described earlier, we have:

$$Q = \{q_0, q_1, q_2, q_3\}$$
$$\Sigma = \{+, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$
$$Q_0 = \{q_0\}$$
$$F = \{q_3\}$$

3.2.2 Transition-table definition of an ϵ-NFA

The transition table for an ϵ-NFA is almost identical to that for an NFA, except there is an extra column to specify the effect of the ϵ-transitions.

For example, the transition-table for the $intRecog$ machine described earlier is:

$intRecog$

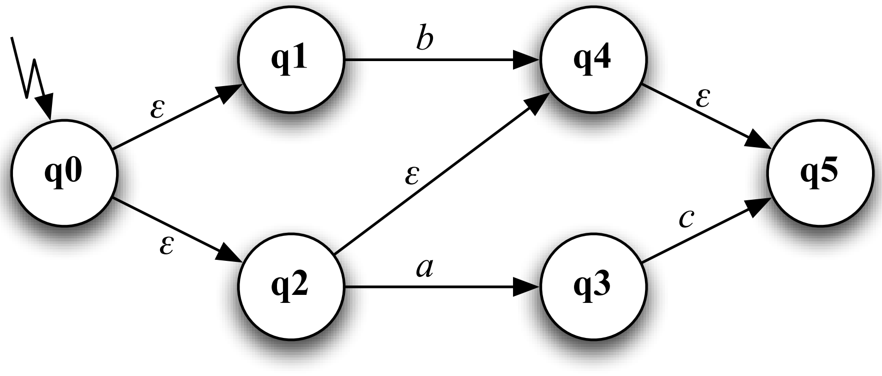
δ	ϵ	$+, -$	$0, 1, \ldots, 9$
$\to q_0$	$\{q_1\}$	$\{q_1\}$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$
q_2	$\{q_3\}$	$\emptyset$	$\{q_2\}$
$*q_3$	$\emptyset$	$\emptyset$	$\emptyset$

3.3 The epsilon-closure

To define the language of an ϵ-NFA, we need to understand a new concept, called the *epsilon-closure*, that is a property of each state. The epsilon-closure of a state q is a set of states. The set includes the state q itself, and all states that can be reached from q, by following *only* ϵ-transitions.

3.3.1 Example of epsilon-closure

Consider the ϵ-NFA shown here:



By looking at the diagram and working backwards, we immediately see:

- $eclose(q_5) = \{q_5\}$
- $eclose(q_4) = \{q_4, q_5\}$, because there is an ϵ-transition from q_4 to q_5
- $eclose(q_3) = \{q_3\}$, because there are no ϵ-transitions leading from state q_3
- $eclose(q_2) = \{q_2, q_4, q_5\}$, because states q_4 and q_5 are reachable from q_2 by taking ϵ-transitions
- $eclose(q_1) = \{q_1\}$, since there are no ϵ-transitions leading from q_1
- $eclose(q_0) = \{q_0, q_1, q_2, q_4, q_5\}$, because we can reach all of these states from q_0, by taking one or more ϵ-transitions

3.3.2 Formal specification of epsilon-closure

We can formally define the epsilon-closure, $eclose(q)$, of a state q like this:

Base case State q is in $eclose(q)$. Formally: $q \in eclose(q)$.

Recurrence case For all states p that are in $eclose(q)$, if state r is reachable from p by an ϵ-transition, then r is in $eclose(q)$. Formally:

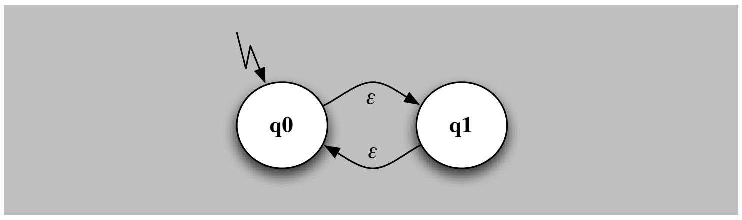
$$\forall p \in eclose(q), \delta(p, \epsilon) \neq \emptyset \text{ and } \delta(p, \epsilon) \subseteq eclose(q)$$

3.3.3 Special cases of epsilon-closure

There are a couple of special cases to worry about.

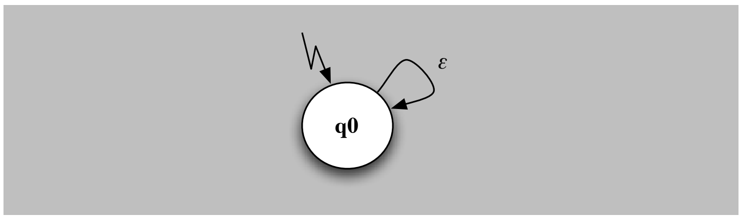
Consider this diagram that has two states connected by ϵ-transitions:

We see that $eclose(q_0) = \{q_0, q_1\}$.



Here is the limiting case of the previous problem:

Once again, there is no problem: $eclose(q_0) = \{q_0\}$.



Thus we see that loops of ϵ-transitions among one or several states cause no problem with the definition of $eclose$.

3.3.4 The extended epsilon-closure

The epilson closure we have defined, $eclose$ takes as parameter a *single* state, and returns a *set of states*.

We will find it convenient to define another function, $ECLOSE$ (note the capital letters!), that takes as parameter a *set of states* and returns a *set of states*. The $ECLOSE$ of a set of states is simply the union of the $eclose$ of each of the individual states, thus the definition is:

$$ECLOSE(Q) = \bigcup_{q \in Q} eclose(q)$$

3.4 The string transition function of an ϵ-NFA

We are interested in knowing what state-set an ϵ-NFA will reach, starting at state-set Q_0, if it is presented with a *string* of symbols $w = s_1 s_2 s_3 \cdots s_n$.

The *string transition function* $\hat{\Delta}$, shows how an ϵ-NFA that starts in any *state-set* P, and receives a sequence of symbols, w, ends up in state-set R.

We can specify the $\hat{\Delta}$ function like this:

$$\hat{\Delta} : \{Q\} \times \Sigma^* \rightarrow \{Q\}$$

(Remember that $\{Q\}$ is the set of all subsets that can be created by selecting states from Q.)

We can define the behaviour of $\hat{\Delta}$ recursively like this:

Base case $\hat{\Delta}(P, \epsilon) = P$ which says that if there is no input, the ϵ-NFA remains in the current state-set P.

Recurrence case We proceed by imagining that we have chopped one symbol, s_1, from the input string, w, and then taking one step forward in the ϵ-NFA machine with that symbol. Finally, starting at the state-set thus reached, we use the extended transition function to process the remainder of the input string, $s_2 s_3 \ldots s_n$. More formally:

1. let $V = ECLOSE(P)$ here V is the set of all states from which the ϵ-NFA could possibly make a transition. It includes the current state-set (P), plus all the states that the machine could reach by taking one or more ϵ-transitions.
2. let $T = \Delta(V, s_1)$ T is the set of all states that will be reached by taking one transition with the symbols_1, starting from the state-set V. (It includes all states *directly* reachable from the current state-set, P, by a transition s_1, *and* all the states *indirectly* reachable by a sequence of epsilon transitions, followed by a transition s_1.) Thus, T is the *next* state-set.

3. We can process the remainder of the input string, (i.e. $s_2, s_3, \ldots, s_n$) by recursively invoking the extended transition function, starting from state-set T: $\hat{\Delta}(T, s_2 s_3 \ldots s_n)$.

By combining the above three steps, we see:

$$\hat{\Delta}(P, s_1 s_2 s_3 \ldots s_n) = \hat{\Delta}(\Delta(ECLOSE(P), s_1), s_2 s_3 \ldots s_n)$$

A recursive evaluation according to these rules *must* terminate in a finite number of steps. Each invocation of $\hat{\Delta}$ occurs with a shorter string of symbols, so we must (eventually) reach the base case: $\hat{\Delta}(P, \epsilon)$, and the recursion will stop.

3.5 The language of an ϵ-NFA

let $E = (Q, \Sigma, \delta, Q_0, F)$ be an ϵ-NFA. We can define $L(E)$, the language of E in a very similar way to that for an NFA. The language is the set of all input strings that take E from its initial state-set Q_0 to a state-set in which *at least one* of the states is an accepting state.

We can find this in three steps:

1. The string transition function, $\hat{\Delta}$ allows us to find the set of states T that E will occupy after processing an input string w:

$$T = \hat{\Delta}(Q_0, w)$$

2. However, there may still be some ϵ-transitionsfrom those states to one (or more) final states. We take account of this by computing, R, the states that are reachable from T by taking only ϵ-transitions:

$$R = ECLOSE(T)$$

3. The string w is accepted by E if at least one of the states in R is in F.

We can compress these three steps into one compact formal statement, like this:

$$L(E) = \{w \in \Sigma^* \mid ECLOSE(\hat{\Delta}(Q_0, w)) \cap F \neq \emptyset\}$$

3.6 Eliminating ϵ-transitions

Given any ϵ-NFA, E, we can find a non-deterministic finite-state automaton (NFA), N, that accepts the same language as E. We do this by eliminating the ϵ-transitions.

Let $E = (Q_E, \Sigma, \delta_E, Q_{0E}, F_E)$ be an ϵ-NFA. We will create an equivalent NFA, $N = (Q_N, \Sigma, \delta_N, q_{0N}, F_N)$, defined like this:

1. N has the same states as E, thus $Q_N = Q_E$. We will see later that sometimes this results in N having unreachable states (states that can never be reached from the initial states). Unreachable states can safely be deleted, thus the number of states in N maybe fewer than in E.

2. E starts in the state-set Q_{0E}, so N will start in the same set of states. Thus:

$$Q_{0N} = Q_{0E}$$

3. The set of final state for E is F_E. It is possible that there are are one or more ϵ-transitions *leading* to from states in Q_E to states in F_E. If so, the states from which those ϵ-transitions come must be regarded as being in the set F_N. Formally:

$$F_D = \{q \in Q_E \mid eclose(q) \in F_E\}$$

4. To compute the transition function, δ_N, we note that for each state q in Q_E, the corresponding transition in N will be expanded to take account of any ϵ-transitionsthat exist from state p. Formally, for any state q in Q_E, we can write:

$$\forall s \in \Sigma : \delta_N(q, s) = \Delta_E(eclose(q), s)$$

This procedure must be applied to *all* the states, q, in Q_E to generate the complete $delta_N$ transition function.

3.7 Example1: Convert integer-recogniser ϵ-NFA to NFA

Earlier we showed an ϵ-NFA that recognised integers. Here is the transition function for that automaton (shown as a table):

$intRecog$

δ	ϵ	$+, -$	$0, 1, \ldots, 9$
$\rightarrow q_0$	$\{q_1\}$	$\{q_1\}$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$
q_2	$\{q_3\}$	$\emptyset$	$\{q_2\}$
$*q_3$	$\emptyset$	$\emptyset$	$\emptyset$

We now convert this ϵ-NFA to a NFA using the method just described.

3.7.1 Compute ϵ-closure

We begin by computing the ϵ-closure, $eclose$, of each state, since we will need this information several times:

state	ϵ-closure
q_0	$\{q_0, q_1\}$
q_1	$\{q_1\}$
q_2	$\{q_2, q_3\}$
q_3	$\{q_3\}$

3.7.2 Compute the starting state-set

The starting state-set of the NFA will be the same as the starting state-set of the ϵ-NFA:

$$\begin{aligned} Q_{0N} &= Q_{0E} \\ &= \{q_0\} \end{aligned}$$

3.7.3 Compute the transition function

We now derive the transition function for *each* state and *each* input symbol.

Let us consider state q_0 first, with the input sybol $+$. From the rules given earlier: $\delta_N(q,s) = \Delta_E(eclose(q),s)$ so:

$$\begin{aligned}
\delta_N(q_0,+) &= \Delta_E(eclose(q_0,+)) \\
&= \Delta_E(\{q_0,q_1\},+) \\
&= \delta_E(q_0,+) \cup \delta_E(q_1,+) \\
&= \{q_1\} \cup \emptyset \\
&= \{q_1\}
\end{aligned}$$

Thus $\delta_N(q_0,+) = \{q_1\}$. It is easy to see that the case for $-$ is the same: $\delta_N(q_0,-) = \{q_1\}$.

Now consider the digits $(0 \ldots 9)$ case, we find:

$$\begin{aligned}
\delta_N(q_0,0) &= \Delta_E(eclose(q_0,0)) \\
&= \Delta_E(\{q_0,q_1\},0) \\
&= \delta_E(q_0,0) \cup \delta_E(q_1,0) \\
&= \emptyset \cup \{q_2\} \\
&= \{q_2\}
\end{aligned}$$

Similar reasoning applies to the other digits. We see $\delta_N(q_0, 0 \ldots 9) = \{q_2\}$.

Thus the first row of the NFA transition table looks like this:

$intRecog$

δ_N	$+,-$	$0,1,\ldots,9$
$\rightarrow q_0$	$\{q_1\}$	$\{q_2\}$

We now repeat this process for state q_1. State-set $\{q_1\}$ is simple to handle, because $eclose\{q_1\}) = \{q_1\}$. If we process a $+/-$ symbol from this state, we die (ie next-state set= $\{\}$), and if we process $0 \ldots 9$, the next-state set is $\{q_2\}$.

Thus the next row of the table shows:

q_1	$\emptyset$	$\{q_2\}$

Similar reasoning applies to state $\{q_2\}$, yielding this row:

q_2	$\emptyset$	$\{q_2\}$

Finally, we process q_3, yielding this row:

q_3	$\emptyset$	$\emptyset$

3.7.4 Compute the final-state set

To determine which states are final states, we must check to see if the ϵ-closureof any state of the NFA contains a final state of the original ϵ-NFA. For each state q in Q_E we evaluate the predicate: $eclose(q) \cap F_E \neq \emptyset$, which will have the value either *false* or *true*.

Working through this for q_0, we find:

Therefore q_0 is not a final state.

$$eclose(q_0) \cap \{q_3\} \neq \emptyset$$
$$\{q_0, q_1\} \cap \{q_3\} \neq \emptyset$$
$$\emptyset \neq \emptyset$$
$$\text{false}$$

Similar reasoning applies to q_1.

For state q_2 we find:

Therefore q_2 *is* a final state.

$$eclose(q_2) \cap \{q_3\} \neq \emptyset$$
$$\{q_2, q_3\} \cap \{q_3\} \neq \emptyset$$
$$\{q_3\} \neq \emptyset$$
$$\text{true}$$

Similar reasoning tells us that q_3 is also a final state. The final-state set is therefore: $\{q_2, q_3\}$.

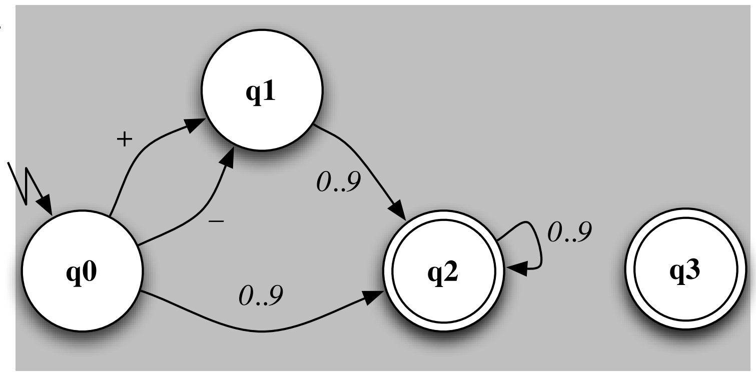
3.7.5 The complete NFA table

Thus the transition-table for the completed NFA is:

$intRecog(NFA)$

δ_N	$+, -$	$0, 1, \ldots, 9$
$\rightarrow q_0$	$\{q_1\}$	$\{q_2\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\{q_2\}$
$*q_3$	$\emptyset$	$\emptyset$

This table corresponds to this NFA diagram:

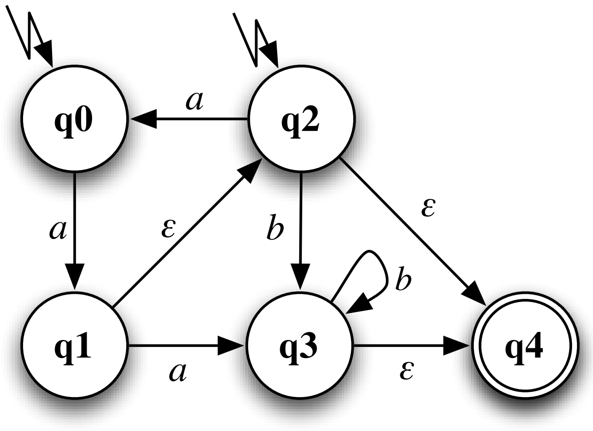


As you can see, it is only slightly more complex than the original ϵ-NFA. After a moment's examination you can also see that:

- It is obviously a correct implementation of the original ϵ-NFA;
- State q_3 is unreachable from any state, and could be deleted; and
- Although we aimed to produce an NFA, this diagram is in fact a *DFA*. (This doesn't always happen — we just got lucky this time.)

3.8 Example2: Convert an ϵ-NFA to an NFA

Here is the diagram of an ϵ-NFA with a lot more ϵ-transitions.



The transition table representation of this ϵ-NFA is:

example2

δ_E	a	b	ϵ
$\rightarrow q_0$	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\{q_3\}$	$\emptyset$	$\{q_2, q_4\}$
$\rightarrow q_2$	$\{q_0\}$	$\{q_3\}$	$\{q_4\}$
q_3	$\emptyset$	$\{q_3\}$	$\emptyset$
$*q_4$	$\emptyset$	$\emptyset$	$\emptyset$

We will now convert this ϵ-NFA to a NFA.

3.8.1 Compute ϵ-closures

We begin by computing the ϵ-closure, *eclose*, of each state, since we will need this information several times:

state	ϵ-closure
q_0	$\{q_0\}$
q_1	$\{q_1, q_2, q_4\}$
q_2	$\{q_2, q_4\}$
q_3	$\{q_3, q_4\}$
q_4	$\{q_4\}$

3.8.2 Compute the starting state-set

The start-states of the NFA will be the epsilon-closure of the start-states of the ϵ-NFA:

$$\begin{aligned} Q_{0N} &= ECLOSE(Q_{0E}) \\ &= ECLOSE(\{q_0, q_2\}) \\ &= eclose(q_0) \cup eclose(q_2) \\ &= \{q_0\} \cup \{q_2\} \\ &= \{q_0, q_2\} \end{aligned}$$

so the NFA will begin in the state-set $\{q_0, q_2\}$.

3.8.3 Compute the transition function

We now derive the transition function for *each* state and *each* input symbol.

Let us consider state q_0 first, with the input sybol a. From the rules given earlier: $\delta_N(q, s) = \Delta_E(eclose(q), s)$

Thus $\delta_N(q_0, a) = \{q_1\}$.

$$\begin{aligned} \delta_N(q_0, a) &= \Delta_E(eclose(q_0, a)) \\ &= \Delta_E(\{q_0\}, a) \\ &= \delta_E(q_0, a) \\ &= \{q_1\} \end{aligned}$$

Now consider the symbol b:

$$\begin{aligned} \delta_N(q_0, b) &= \Delta_E(eclose(q_0, b) \\ &= \Delta_E(\{q_0\}, b) \\ &= \delta_E(q_0, b) \\ &= \emptyset \end{aligned}$$

Thus the first row of the NFA transition table looks like this:

$example2$

δ_N	a	b
q_0	$\{q_1\}$	$\emptyset$

We repeat this process for state q_1, with input a:

$$\begin{aligned} \delta_N(q_1, a) &= \Delta_E(eclose(q_1, a) \\ &= \Delta_E(\{q_1, q_2, q_4\}, a) \\ &= \delta_E(q_1, a) \cup \delta_e(q_2, a) \cup \delta_E(q_4, a) \\ &= \{q_3\} \cup \{q_0\} \cup \emptyset \\ &= \{q_0, q_3\} \end{aligned}$$

And for input b:

$$\begin{aligned} \delta_N(q_1, b) &= \Delta_E(eclose(q_1, b) \\ &= \Delta_E(\{q_1, q_2, q_4\}, b) \\ &= \delta_E(q_1, b) \cup \delta_e(q_2, b) \cup \delta_E(q_4, b) \\ &= \emptyset \cup \emptyset \cup \emptyset \\ &= \emptyset \end{aligned}$$

Thus the next row shows:

q_1	$\{q_0, q_3\}$	$\emptyset$

Processing q_2 with a, we find:

$$\begin{aligned}\delta_N(q_2,a) &= \Delta_E(eclose(q_2,a)\\ &= \Delta_E(\{q_2,q_4\},a)\\ &= \delta_E(q_2,a)\cup\delta_E(q_4,a)\\ &= \{q_0\}\cup\emptyset\\ &= \{q_0\}\end{aligned}$$

and with b:

$$\begin{aligned}\delta_N(q_2,b) &= \Delta_E(eclose(q_2,b)\\ &= \Delta_E(\{q_2,q_4\},b)\\ &= \delta_E(q_2,b)\cup\delta_E(q_4,b)\\ &= \{q_3\}\cup\emptyset\\ &= \{q_3\}\end{aligned}$$

So the next row of the table is:

$$q_2 \;\|\; \{q_0\} \;|\; \{q_3\}$$

For q_3 with a we get:

$$\begin{aligned}\delta_N(q_3,a) &= \Delta_E(eclose(q_3,a)\\ &= \Delta_E(\{q_3,q_4\},a)\\ &= \delta_E(q_3,a)\cup\delta_e(q_4,a)\\ &= \emptyset\cup\emptyset\\ &= \emptyset\end{aligned}$$

And with b:

$$\begin{aligned}\delta_N(q_3,b) &= \Delta_E(eclose(q_3,b)\\ &= \Delta_E(\{q_3,q_4\},b)\\ &= \delta_E(q_3,b)\cup\delta_e(q_4,b)\\ &= \{q_3\}\cup\emptyset\\ &= \{q_3\}\end{aligned}$$

So the fourth row of the table is:

$$q_3 \;\|\; \emptyset \;|\; \{q_3\}$$

And finally for q_4 with a:

$$\begin{aligned}\delta_N(q_4,a) &= \Delta_E(\mathit{eclose}(q_4,a)\\ &= \Delta_E(\{q_4\},a)\\ &= \delta_E(q_4,a)\\ &= \emptyset\end{aligned}$$

And with b:

$$\begin{aligned}\delta_N(q_4,b) &= \Delta_E(\mathit{eclose}(q_4,b)\\ &= \Delta_E(\{q_4\},b)\\ &= \delta_E(q_4,b)\\ &= \emptyset\end{aligned}$$

So the last row of the table is:

$$q_4 \parallel \emptyset \mid \emptyset$$

3.8.4 Compute the final-state set

We now determine if any of the states, q, are final-states, by evaluating the predicate: $\mathit{eclose}(q) \cap F_E \neq \emptyset$

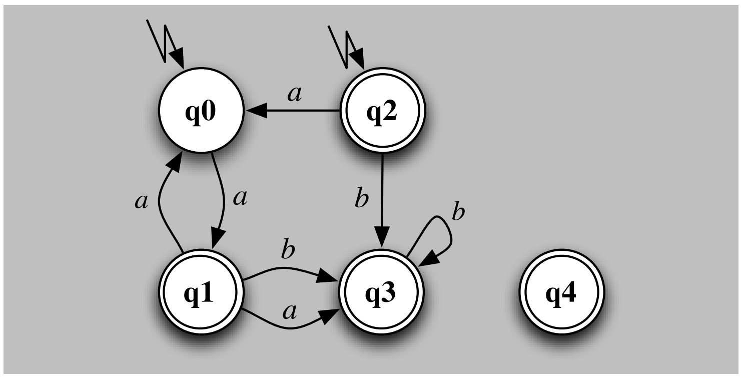
$$\begin{aligned}\mathit{eclose}(q_0) \cap \{q_4\} \neq \emptyset &\Rightarrow \text{false}\\ \mathit{eclose}(q_1) \cap \{q_4\} \neq \emptyset &\Rightarrow \text{true}\\ \mathit{eclose}(q_2) \cap \{q_4\} \neq \emptyset &\Rightarrow \text{true}\\ \mathit{eclose}(q_3) \cap \{q_4\} \neq \emptyset &\Rightarrow \text{true}\\ \mathit{eclose}(q_4) \cap \{q_4\} \neq \emptyset &\Rightarrow \text{true}\end{aligned}$$

3.8.5 The complete transition table

Here is the complete table and corresponding diagram:

$example2(NFA)$

δ_N	a	b
$\rightarrow q_0$	$\{q_1\}$	$\emptyset$
$*q_1$	$\{q_1\}$	$\{q_3\}$
$\rightarrow q_2$	$\{q_1\}$	$\{q_3\}$
$*q_3$	$\emptyset$	$\{q_3\}$
$*q_4$	$\emptyset$	$\emptyset$



3.9 Lazy/Greedy evaluation

The procedures we have so far described for eliminating ϵ-transitions can be described as *lazy evaluation* — the machine does not evaluate an ϵ-transitionuntil a real symbol arrives and needs to be processed.

There is an alternative view: whenever we have an opportunity to execute an ϵ-transition, the machine could immediately take it. This approach is known as *greedy evaluation*.

The difference between the two approaches is quite small — it depends on when the ϵ-closureoperation is performed.

With *lazy* evaluation, we have seen that we perform the ϵ-closure operation *before* performing the $\Delta(q, s)$ step to process a symbol s. Lazy evaluation requires us to do a final ϵ-closureoperation before deciding whether the machine has reached an accepting state.

With *greedy* evaluation, the order of operations is reversed: after performing the $\Delta(q, s)$ step to process a symbol s, the machine executes an ϵ-closureoperation, to "rush ahead". Greedy evaluation requires us to perform an ϵ-closure operation at the *beginning*, to "rush ahead" after the machine has been initalised.

3.10 Converting the intRecog ϵ-NFAto a NFA by greedy evaluation

3.10.1 Compute ϵ-closures

We begin by computing the ϵ-closure, $eclose$, of each state, since we will need this information several times:

state	ϵ-closure
q_0	$\{q_0, q_1\}$
q_1	$\{q_1\}$
q_2	$\{q_2, q_3\}$
q_3	$\{q_3\}$

3.10.2 Compute the starting state-set

The start-states of the NFA will be the ϵ-closureof the start-states of the ϵ-NFA:

$$\begin{aligned} Q_{0N} &= ECLOSE(Q_{0E}) \\ &= ECLOSE(\{q_0\}) \\ &= eclose(q_0) \\ &= \{q_0, q_1\} \end{aligned}$$

3.10.3 Compute the transition function

We now derive the transition function for *each* state and *each* input symbol.

Let us consider state q_0 first, with the input sybol $+$. With greedy eavluation, we the order of evaluation is the reverse of lazy evaluation: $\delta_N(q,s) = ECLOSE(\delta_E(q,s))$ so:

$$\begin{array}{rcl} \delta_N(q_0,+) & = & ECLOSE(\delta_E(q_0,+)) \\ & = & ECLOSE(\{q_1\}) \\ & = & eclose(q_1) \\ & = & \{q_1\} \end{array}$$

Thus $\delta_N(q_0,+) = \{q_1\}$. It is easy to see that the case for $-$ is the same: $\delta_N(q_0,-) = \{q_1\}$.

Now consider the digits $(0\ldots9)$ case, we find:

Similar reasoning applies to the other digits. We see $\delta_N(q_0, 0\ldots9) = \{q_2\}$.

$$\begin{array}{rcl} \delta_N(q_0,0) & = & ECLOSE(\delta_E(q_0,0)) \\ & = & ECLOSE(\emptyset) \\ & = & \emptyset \end{array}$$

Thus the first row of the NFA transition table looks like this:

$intRecog$

δ_N	$+,-$	$0,1,\ldots,9$
q_0	$\{q_1\}$	$\emptyset$

We now repeat this process for state q_1, with $+/-$ symbols:

$$\begin{array}{rcl} \delta_N(q_1,+) & = & ECLOSE(\delta_E(q_1,+)) \\ & = & ECLOSE(\emptyset) \\ & = & \emptyset \end{array}$$

And for $0\ldots9$:

$$\begin{array}{rcl} \delta_N(q_1,0) & = & ECLOSE(\delta_E(q_1,0)) \\ & = & ECLOSE(\{q_2\}) \\ & = & eclose(q_2) \\ & = & \{q_2,q_3\} \end{array}$$

Thus the next row of the table shows:

q_1	$\emptyset$	$\{q_2,q_3\}$

Repeating this for state q_2, with $+/-$ symbols:

$$\begin{array}{rcl} \delta_N(q_2,+) & = & ECLOSE(\delta_E(q_2,+)) \\ & = & ECLOSE(\emptyset) \\ & = & \emptyset \end{array}$$

And for $0 \ldots 9$:

$$\begin{aligned}\delta_N(q_1, 0) &= ECLOSE(\delta_E(q_1, 0)) \\ &= ECLOSE(\{q_2\}) \\ &= eclose(q_2) \\ &= \{q_2, q_3\}\end{aligned}$$

Thus the next row of the table is:

$$q_2 \parallel \emptyset \mid \{q_2, q_3\}$$

Finally, we process q_3, yielding this row:

$$q_3 \parallel \emptyset \mid \emptyset$$

3.10.4 Compute the final-state set

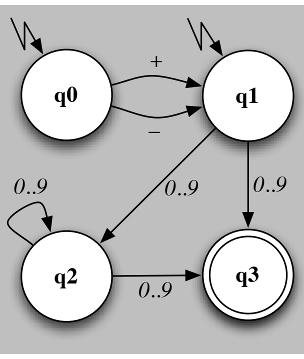
With greedy evaluation, the final state-set of the NFA is exactly the same as that of the ϵ-NFA. Thus $f_N = f_E = \{q_3\}$.

3.10.5 The complete NFA table

Combining the pieces, the transition-table for the intRecog NFA, and the corresponding diagram is:

$intRecog(NFA)greedy$

δ_N	$+, -$	$0, 1, \ldots, 9$
$\rightarrow q_0$	$\{q_1\}$	$\emptyset$
$\rightarrow q_1$	$\emptyset$	$\{q_2, q_3\}$
q_2	$\emptyset$	$\{q_2, q_3\}$
$*q_3$	$\emptyset$	$\emptyset$



We leave the task of converting the $example2$ ϵ-NFA to an NFA by greedy evaluation as an exercise.