

1.1.1 Minimization of DFAs

- Two automata that recognize exactly the same language are *equivalent* with each other
- A finite automaton is *minimal* if it has the smallest number of states among equivalent automata
- An automaton that has more states than in an equivalent minimal automaton is called *redundant*
- Algorithms producing automata do not always generate a minimal automaton
- Handling a minimal automaton is more efficient than that of a redundant automaton



Algorithm MINIMIZE

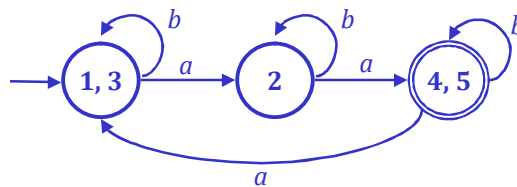
Input: DFA $M = (Q, \Sigma, \delta, q_0, F)$.

1. Remove all states of M that are unreachable from the start state.
2. Construct the following undirected graph G whose nodes are the states of M .
3. Place an edge in G connecting every accept state with every nonaccept state. Add additional edges as follows.
4. Repeat until no new edges are added to G :
 1. For every pair $q, r \in Q, q \neq r$, and every $a \in \Sigma$: add the edge (q, r) to G if $(\delta(q, a), \delta(r, a))$ is an edge of G .
 2. For each state $q \in Q$ let $[q]$ be the collection of states
 2. $[q] = \{q\} \cup \{r \in Q \mid \text{no edge joins } q \text{ and } r \text{ in } G\}$.
5. Form a new DFA $M' = (Q', \Sigma, \delta', q'_0, F')$, where
 - $Q' = \{[q] \mid q \in Q\}$, (removing doubles)
 - $\delta'([q], a) = [\delta(q, a)]$, for every $q \in Q$ and $a \in \Sigma$,
 - $q'_0 = [q_0]$ and
 - $F' = \{[q] \mid q \in F\}$.
6. Output M' .



The End Result

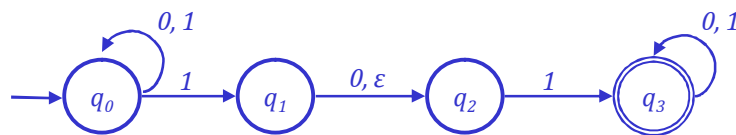
- An automaton M' that is equivalent with the input automaton M , such that it has the minimum number of states
- Automaton M' is unique (up to the naming of the states).



1.2 Nondeterministic Finite Automata (NFAs)

- In an NFA a state can have many possible alternative transitions with the same symbol of the alphabet
- Also ϵ -transitions are allowed
- Implementing nondeterministic behavior is not straightforward (though possible), but as a modeling tool it is quite useful
- Via NFAs we can connect DFAs and *regular expressions*



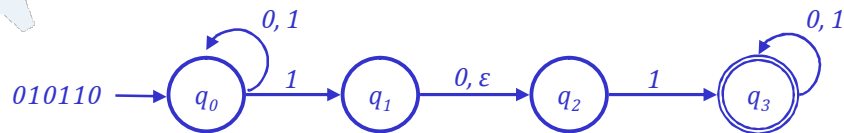


- The definition of an automaton requires the transition function to be a *function*
- On the other hand, in an NFA the transition function should get mapped to a *set* of values
- An NFA accepts a string if a sequence of possible states leads to a final state.
 - Only if no such sequence exists will the NFA reject the input string
- E.g. the previous NFA accepts the string **010110** because it can be processed as follows

$$(q_0, 010110) \ni (q_0, 10110) \ni (q_1, 0110)$$

$$(q_2, 110) \ni (q_3, 10) \ni (q_3, 0) \ni (q_3, \epsilon)$$





- On the other hand, we can end up in a rejecting state:

$$\begin{aligned} (q_0, 010110) &\ni (q_0, 10110) \ni (q_0, 0110) \\ &\ni (q_0, 110) \ni (q_0, 10) \ni (q_1, 0) \ni (q_2, \varepsilon) \end{aligned}$$



Definition of an NFA

- Let $\mathcal{P}(A) = \{ B \mid B \subseteq A \}$ denote the **power set** of the set A and for an alphabet Σ : $\Sigma_\varepsilon = \Sigma \cup \{ \varepsilon \}$
- A nondeterministic finite automaton is a 5-tuple $N = (Q, \Sigma, \delta, q_0, F)$
 - Q is a finite set of **states**,
 - Σ is a finite **alphabet**,
 - $\delta: Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)$ is the (set-valued) **transition function**, that also allows ε -transitions
 - $q_0 \in Q$ is the **start state**, and
 - $F \subseteq Q$ is the set of (accepting) **final states**



- The transition function of the previous automaton is

	0	1	ϵ
$\rightarrow q_0$	$\{q_0\}$	$\{q_0q_1\}$	\emptyset
q_1	$\{q_2\}$	\emptyset	$\{q_2\}$
q_2	\emptyset	$\{q_3\}$	\emptyset
$\leftarrow q_3$	$\{q_3\}$	$\{q_3\}$	\emptyset

- Now we can easily express the error state as an empty set of possible next states



- An NFA $N = (Q, \Sigma, \delta, q_0, F)$ accepts the string w ,
 - If we can write it as $w = y_1y_2\dots y_m \in \Sigma_\epsilon^m$ and a sequence of states r_0, r_1, \dots, r_m exists in Q s.t.
 - $r_0 = q_0$,
 - $r_{i+1} \in \delta(r_i, y_{i+1})$, $i = 0, \dots, m-1$, and
 - $r_m \in F$.
- DFAs are a special case of NFAs \rightarrow
all languages that can be recognized using the former can also be recognized using the latter
- Also the other way around: *DFAs and NFAs recognize the same set of languages*



Theorem 1.39 Let $A = L(N)$ be the language recognized by some NFA N . There exists a DFA M such that $L(M) = A$

Proof. Let $N = (Q, \Sigma, \delta, q_0, F)$. We construct a DFA $M = (Q', \Sigma, \delta', q_0', F')$ that simulates the computation of N in parallel in all its possible states at all times. Let us first consider the easier situation where N has no ϵ arrows.

Every state of M is a set of states of N

$$Q' = \mathcal{P}(Q)$$

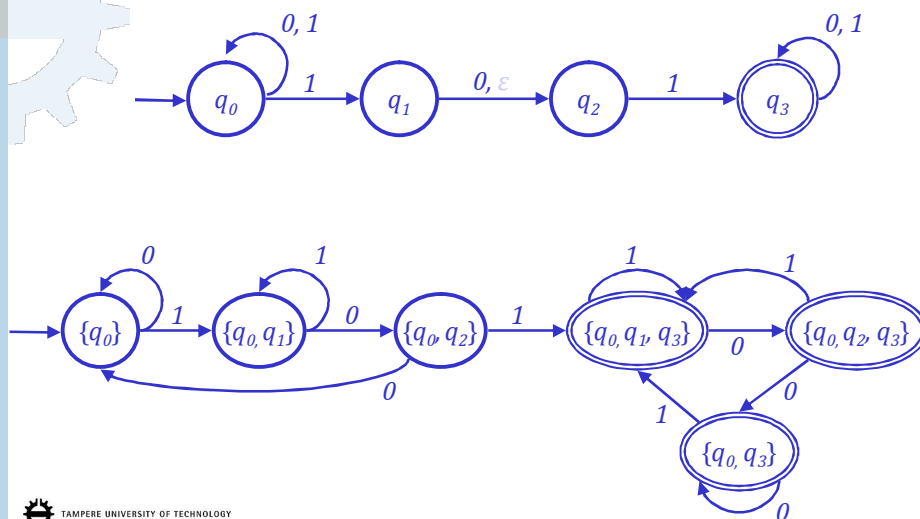
$$q_0' = \{q_0\}$$

$$F' = \{R \in Q' \mid R \text{ contains an accept state } r \in F\}$$

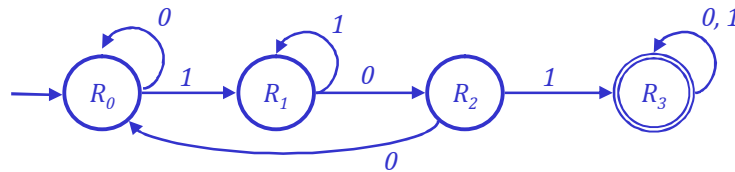
$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$



Without ϵ arrows



After Minimization



Let us check that $L(M) = L(N)$. The equivalence of the languages follows when we prove for all $x \in \Sigma^*$ and $r \in Q$ that

$$(q_0, x) \ggg_N(r, \varepsilon) \Leftrightarrow (\{q_0\}, x) \ggg_M(R, \varepsilon) \text{ and } r \in R,$$

where the notation $(q_0, x) \ggg_N(r, \varepsilon)$ means that in automaton N we can process the string x starting from state q_0 so that we end up in state r and there are no more symbols to process (ε).

We prove it using induction over the length of the string x :

1. **Basis:** $|x| = 0$: $(q_0, \varepsilon) \ggg_N(r, \varepsilon) \Leftrightarrow r = q_0$.
Similarly $(\{q_0\}, \varepsilon) \ggg_M(R, \varepsilon) \Leftrightarrow R = \{q_0\}$



2. *Induction hypothesis*: the claim holds when $|x| \leq k$
3. $|x| = k+1$: Then $x = ya$ for some y , $|y| = k$, for which the claim holds by the induction hypothesis. Now,

$$(q_\emptyset x) = (q_\emptyset ya) \ggg_N(r, \epsilon)$$

$$\Leftrightarrow \exists r' \in Q \text{ s.t. } (q_\emptyset ya) \ggg_N(r', a) \text{ and } (r', a) \succ_N(r, \epsilon)$$

$\succ =$ in one transition

$$\Leftrightarrow \exists r' \in Q \text{ s.t. } (q_\emptyset y) \ggg_N(r', \epsilon) \text{ and } (r', a) \succ_N(r, \epsilon)$$

By induction hypothesis we get

$$\Leftrightarrow \exists r' \in Q \text{ s.t. } (\{q_\emptyset\}, y) \ggg_M(R', \epsilon) \text{ and } r' \in R' \text{ and } r \in \delta(r', a)$$

Rearranging yields

$$\Leftrightarrow (\{q_\emptyset\}, y) \ggg_M(R', \epsilon) \text{ and } \exists r' \in R' \text{ s.t. } r \in \delta(r', a)$$

By the definition of the transition function δ'

$$\Leftrightarrow (\{q_\emptyset\}, y) \ggg_M(R', \epsilon) \text{ and } r \in \boxtimes_{r' \in R'} \delta(r', a) = \delta'(R', a)$$

Let us return a and name $\delta'(R', a)$

$$\Leftrightarrow (\{q_\emptyset\}, ya) \ggg_M(R', a) \text{ and } r \in \delta'(R', a) = R$$

$$\Leftrightarrow (\{q_\emptyset\}, ya) \ggg_M(R', a) \text{ and } (R', a) \succ_M(R, \epsilon) \text{ and } r \in R$$

Concluding

$$\Leftrightarrow (\{q_\emptyset\}, x) = (\{q_\emptyset\}, ya) \ggg_M(R, \epsilon) \text{ and } r \in R$$

Which completes the proof of the claim

- In order to take the ϵ arrows into account, we compute for each state $R \subseteq Q$ of M the collection of states that can be reached from R by going only along ϵ arrows:

$$E(R) = \{ q \mid q \text{ can be reached from } R \text{ by traveling along } 0 \text{ or more } \epsilon \text{ arrows} \}$$

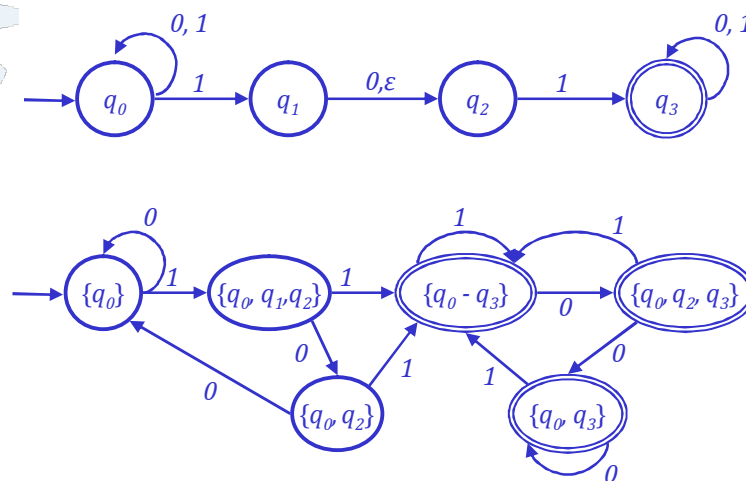
- It is enough to modify the transition function of M and start state to take the ϵ arrows into account:

$$\begin{aligned} \delta'(R, a) &= \cup_{r \in R} E(\delta(r, a)) \\ q_0' &= E(\{q_0\}) \end{aligned}$$

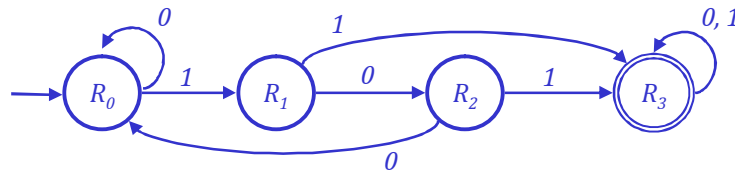
□



With ϵ arrows



After Minimization



Theorem 1.45 *The class of regular languages is closed under the union operation.*

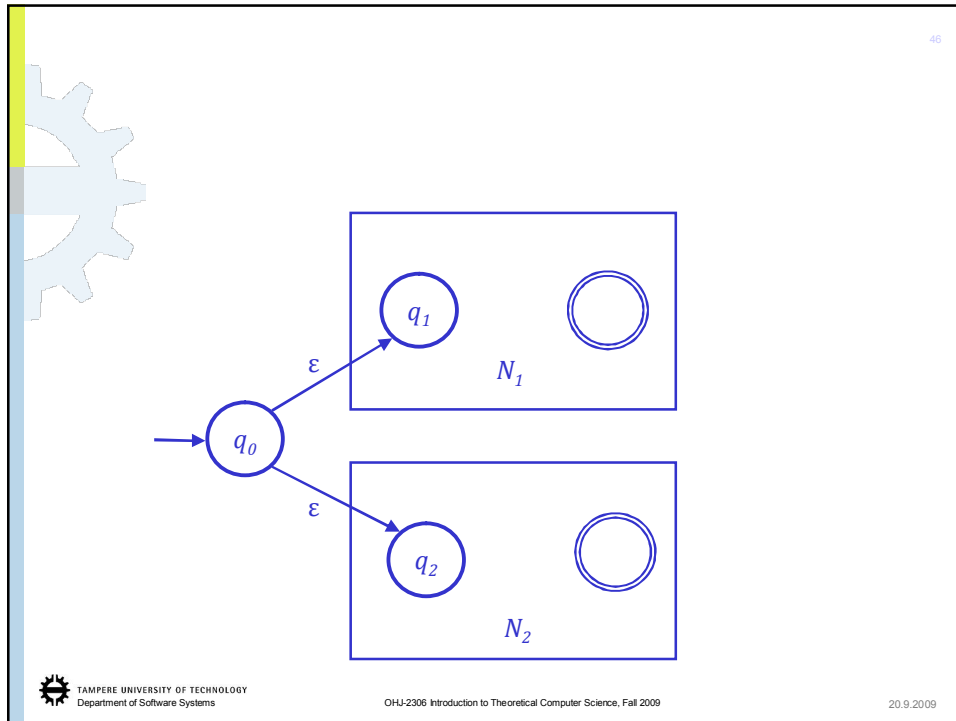
Proof. Let the languages A_1 and A_2 be regular. Then, there exists (nondeterministic) finite automata $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, which recognize these two languages.

Let us construct an automaton $N = (Q, \Sigma, \delta, q_0, F)$ for recognizing the language $A_1 \cup A_2$.

- $Q = \{q_0\} \cup Q_1 \cup Q_2$,
- The start state of N is q_0 ,
- $F = F_1 \cup F_2$ and

$$\delta(q, a) = \begin{cases} \delta_1(q, a), & q \in Q_1 \\ \delta_2(q, a), & q \in Q_2 \\ \{q_1, q_2\}, & q = q_0 \text{ and } a = \varepsilon \\ \emptyset, & q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$





Theorem 1.47 *The class of regular languages is closed under the concatenation operation.*

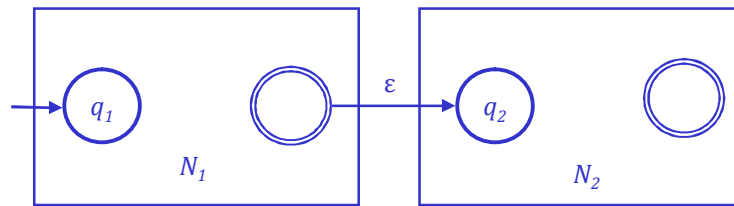
Proof. Let the languages A_1 and A_2 be regular. Then, there exists (nondeterministic) finite automata $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, which recognize these two languages.

Let us construct an automaton $N = (Q, \Sigma, \delta, q_1, F_2)$ for recognizing $A_1 \circ A_2$.

- $Q = Q_1 \cup Q_2$,
- The start state of N is q_1 ,
- The final states of N are those in F_2 and

$$\delta(q, a) = \begin{cases} \delta_1(q, a), & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a), & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_2\}, & q \in F_1 \text{ and } a = \epsilon \\ \delta_2(q, a), & q \in Q_2 \end{cases}$$

□



Theorem 1.49. *The class of regular languages is closed under the star operation.*

Proof. Let the language A be regular. Then, there exists a (nondeterministic) finite automaton $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, which recognizes the language.

Let us construct an automaton $N = (Q, \Sigma, \delta, q_0, F)$ for recognizing A^* .

- $Q = \{q_0\} \cup Q_1$,
- The new start state of N is q_0 ,
- $F = \{q_0\} \cup F_1$ and

$$\delta(q, a) = \begin{cases} \delta_1(q, a), & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a), & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_1\}, & q \in F_1 \text{ and } a = \epsilon \\ \{q_1\} & q = q_0 \text{ and } a = \epsilon \\ \emptyset, & q = q_0 \text{ and } a \neq \epsilon \end{cases}$$

□



