NFA

Asai Asaithambi

Spring 2018

Topic Overview

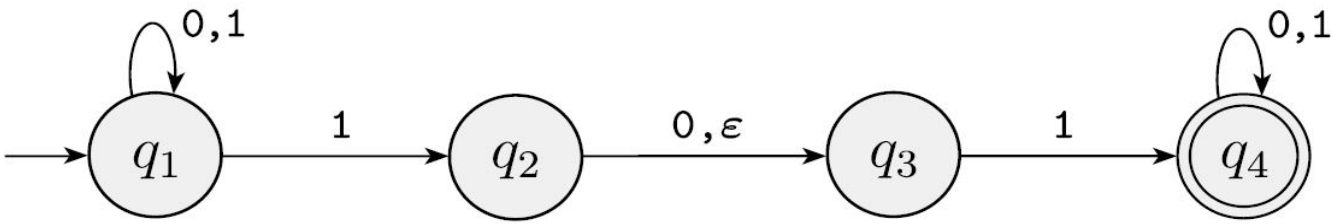
What does this topic cover?

- Nondeterminism
  - Next State(s)
  - Transition labels
- NFA to equivalent DFA conversion
- Regular languages closed under regular operations

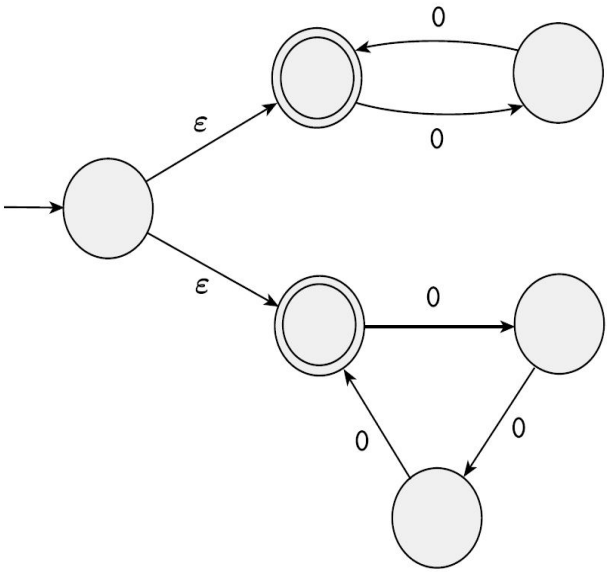
Determinism vs. Nondeterminism

DETERMINISM	NONDETERMINISM
Unique next state	May be multiple next states
One edge for one label	May be multiple edges for same label
No ε transitions	ε transitions allowed
No choices for next state	Next state chosen at random
Only one state to consider for processing strings	All possible next states considered in parallel

An Example NFA: N_1



Another NFA Example

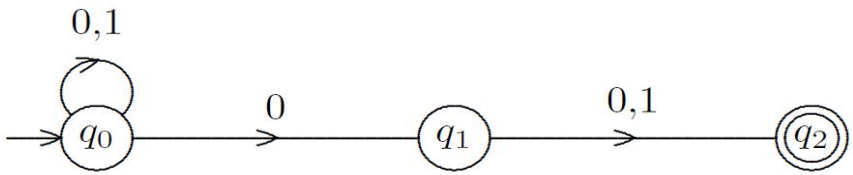


This NFA accepts strings of the form 0^k
where k is even or a multiple of 3.

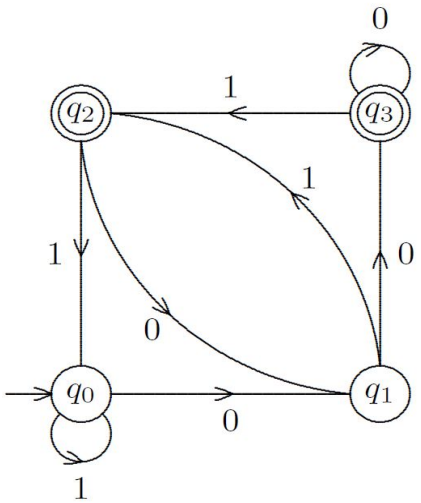
It accepts ε, 00, 000,
but does not accept 0 or 00000.

Every NFA has an equivalent DFA

The following NFA accepts
$\{w \in \{0,1\}^* \mid w$ has a 0 in the second to last position$\}$.



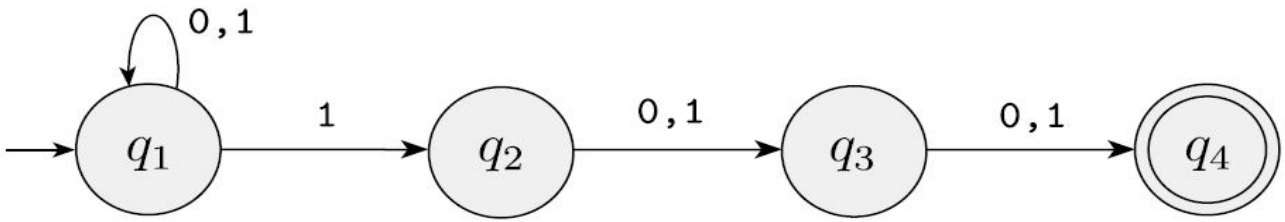
The equivalent DFA is:



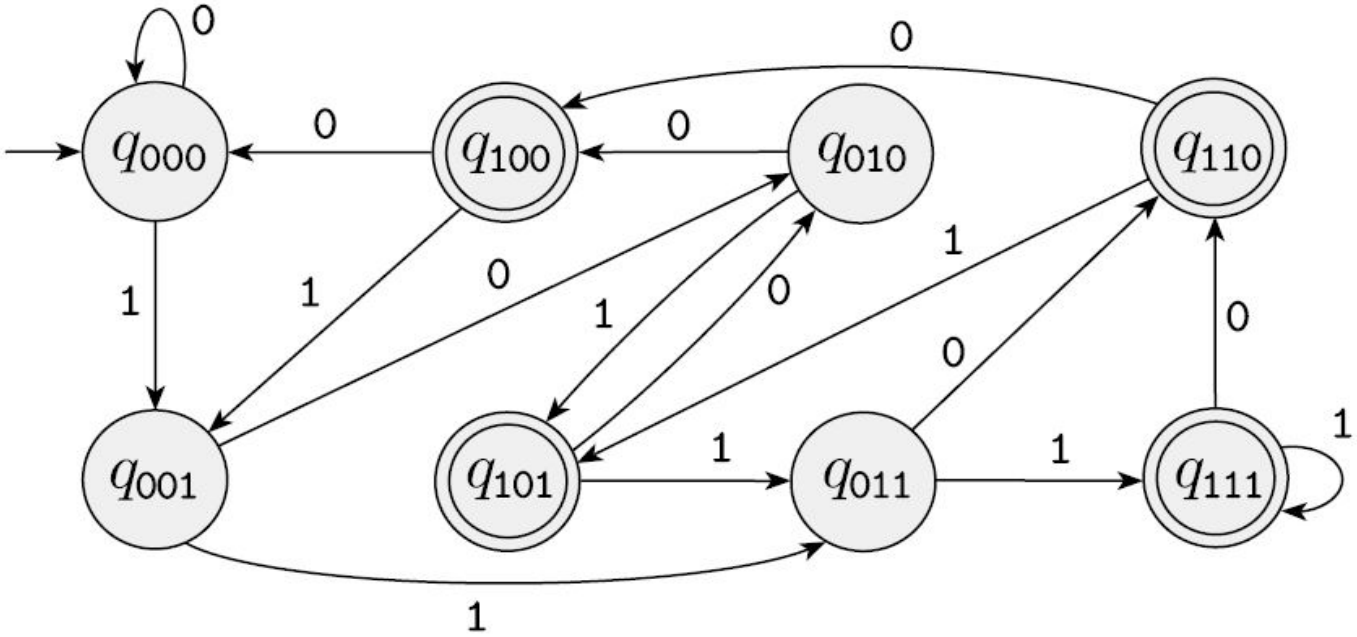
Generally, the equivalent DFAs are large and hard to construct.

Another equivalent NFA–DFA Pair

NFA



DFA



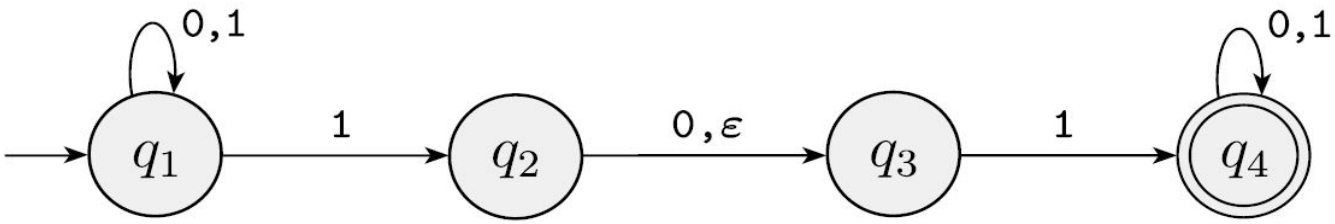
Formal Definition of NFA

A ***nondeterministic finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

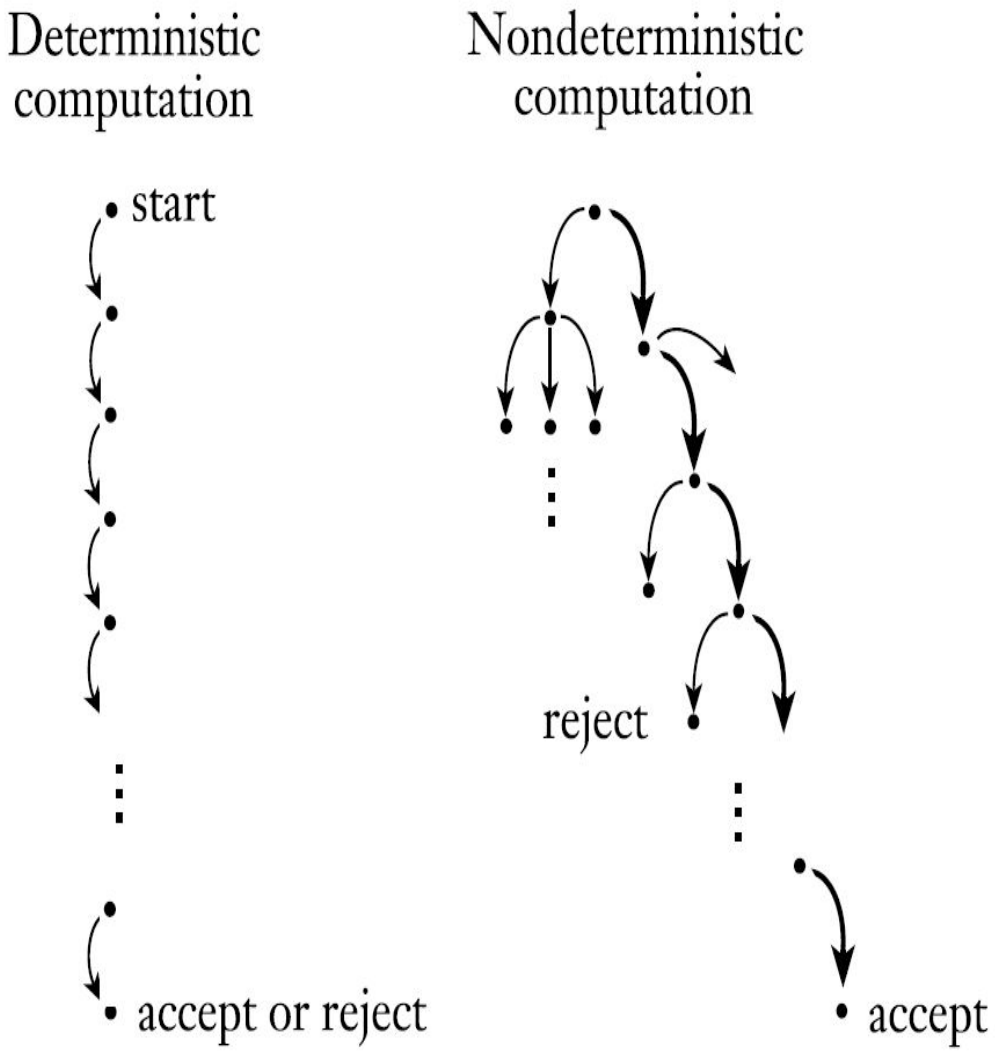
Note: $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ and $\mathcal{P}(Q) =$ power set of Q

Formal Definition of N_1

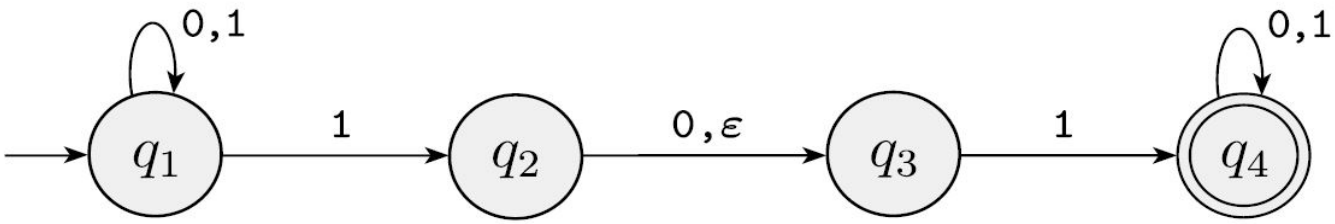


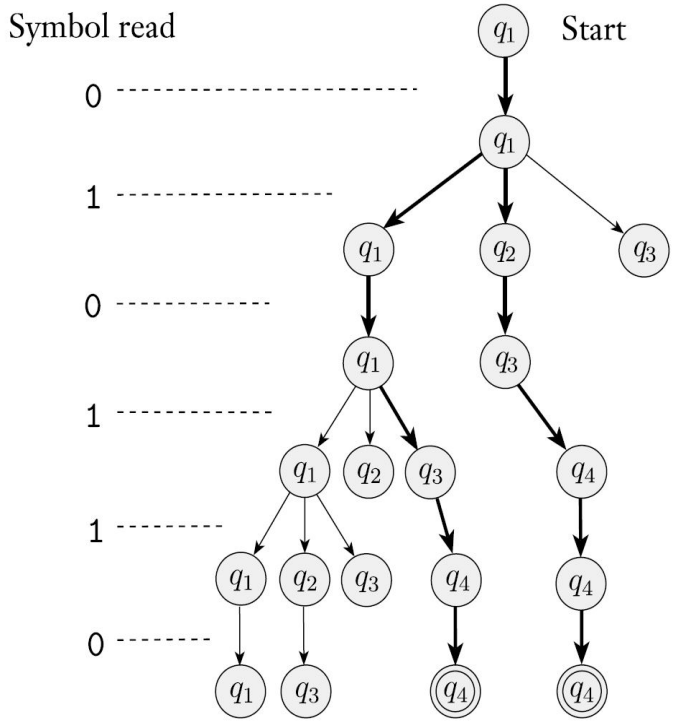
	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

DFA vs. NFA Computation



Processing of 010110 on N_1





Formal Definition of NFA Computations

- Let NFA N be defined by $N = (Q, \Sigma, \delta, q_0, F)$
- How does N process the string $w = w_1 w_2 \cdots w_n,\ w_i \in \Sigma_\varepsilon$?
- N starts at state q_0 and reads input w_1
- It reaches one or more of the next states based on δ
- At each such next state, it reads the next input w_2
- In this manner, it traces several sequences of states
- $r_0, r_1, \cdots, r_n \in Q$ such that:
  - $r_0 = q_0$
  - $r_{i+1} \in \delta(r_i, w_{i+1}),\ i = 0, 1, 2, \cdots, n-1$
- w is accepted, or $w \in L(N)$ if $r_n \in F$
- w is not accepted, or $w \notin L(N)$ if $r_n \notin F$

Equivalence of NFAs and DFAs

- Every NFA has an equivalent DFA.
- If the NFA has k states, then there are 2^k subsets.
- Each of the 2^k subsets will be state in the equivalent DFA.
- Let N be the given NFA, with $N = (Q, \Sigma, \delta, q_0, F)$.
- We wish to construct the DFA $M = (Q', \Sigma, \delta', q_0', F')$.
- Express Q', δ', q_0', and F' in terms of Q, δ, q_0, and F.
- We have already hinted that $Q' = \mathcal{P}(Q)$.
- If there are no ε arrow, then $q_0' = \{q_0\}$.
- If there are no ε arrows, we could also write

$$F' = \{R \in Q' \mid R \text{ contains an acccept state in } N\}$$

Equivalence of NFAs and DFAs (cont.)

- To take into account the effect of ε arrows, for $R \subseteq Q$, we define:

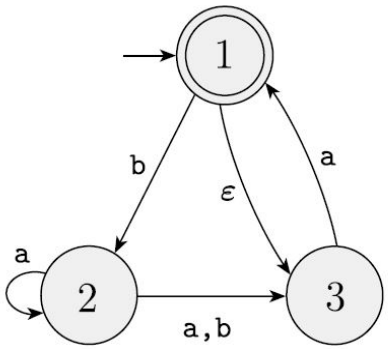
$$E(R) = \{q \mid q \text{ can be reached from } R \text{ through 0 or more } \varepsilon \text{ arrows}\}.$$

- With this definition, we obtain:

$$q_0' = E(\{q_0\}),$$
$$\delta'(R, a) = \{q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R\}.$$

NFA to DFA Conversion Example

Given the below NFA N, we construct its equivalent DFA D:



D's state set is:

$$\{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$$

N's start state is 1.

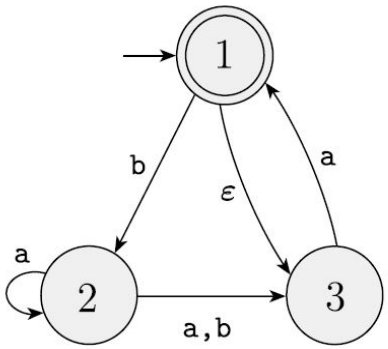
Thus, D's start state is:

$$E(\{1\}) = \{1,3\}.$$

D's final states are any of the subsets containing one or more of the final states of N.

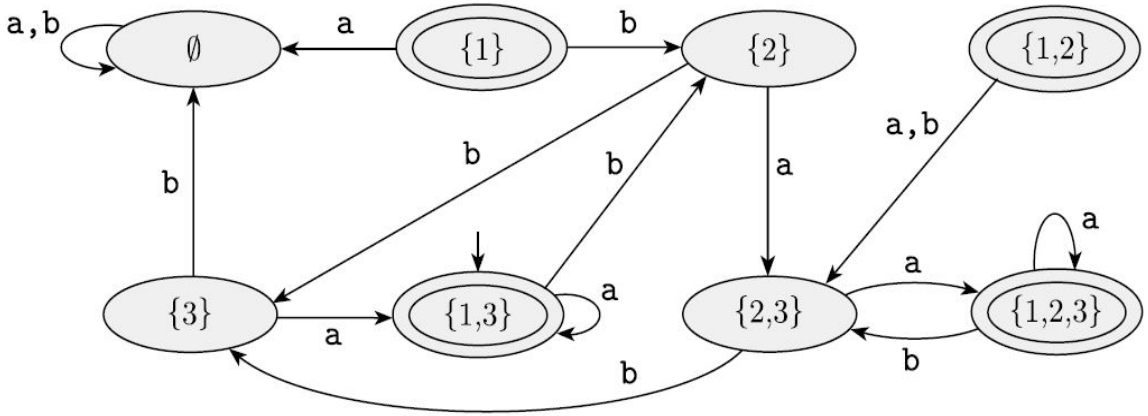
Therefore D's final states are: $\{1\}$, $\{1,2\}$, $\{1,3\}$, and $\{1,2,3\}$.

NFA to DFA Conversion: Transition Function

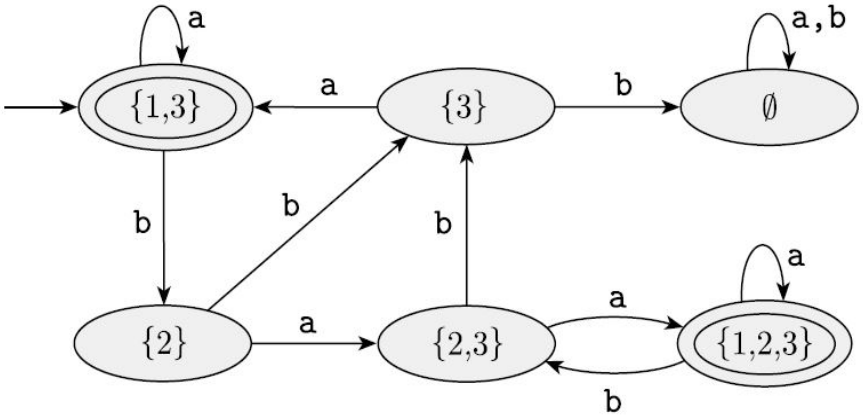


	a	b
$\{1\}$	$\emptyset$	$\{2\}$
$\{2\}$	$\{2,3\}$	$\{3\}$
$\{3\}$	$\{1,3\}$	$\emptyset$
$\{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{1,3\}$	$\{1,3\}$	$\{2\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\emptyset$	$\emptyset$	$\emptyset$

NFA to DFA: State Diagram

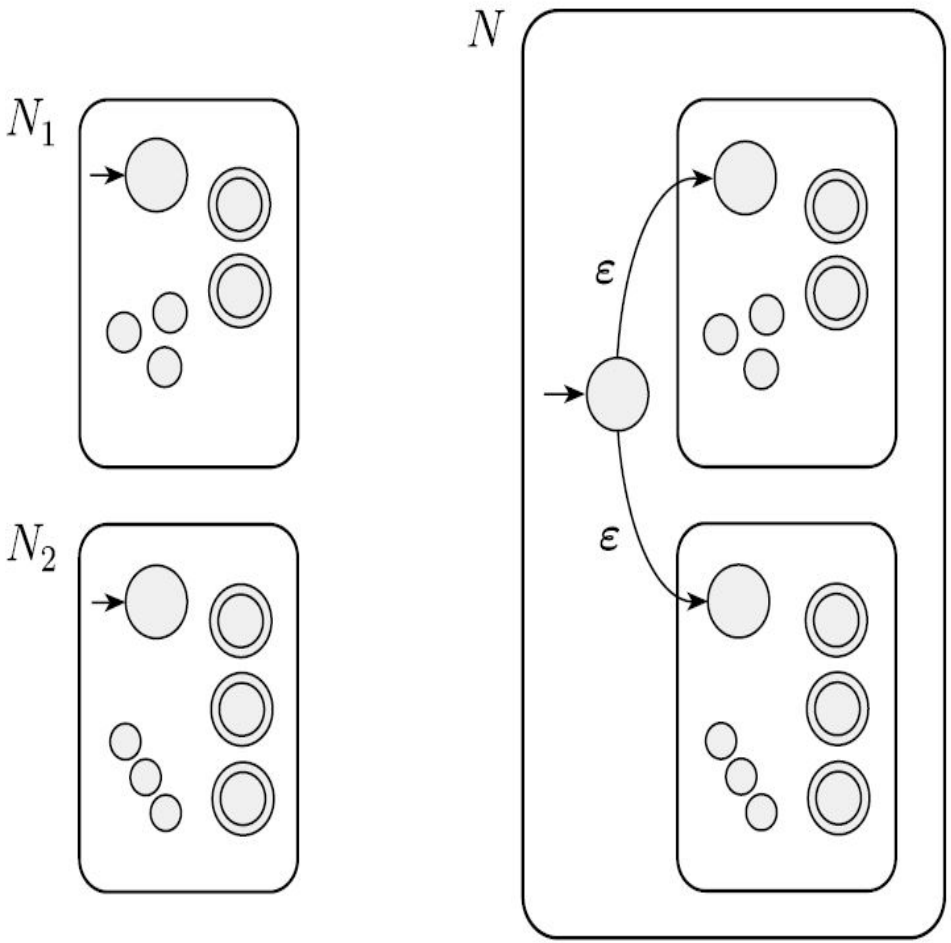


Note that $\{1\}$ and $\{1, 2\}$ are unreachable – remove them.



A language is regular if and only if some NFA recognizes it.

Closure Under Union

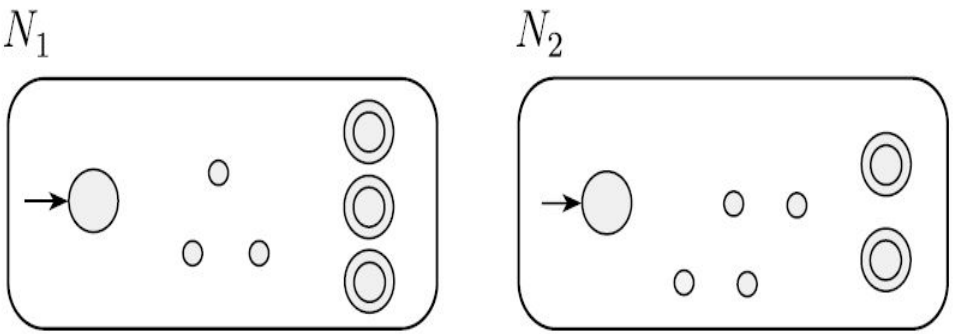


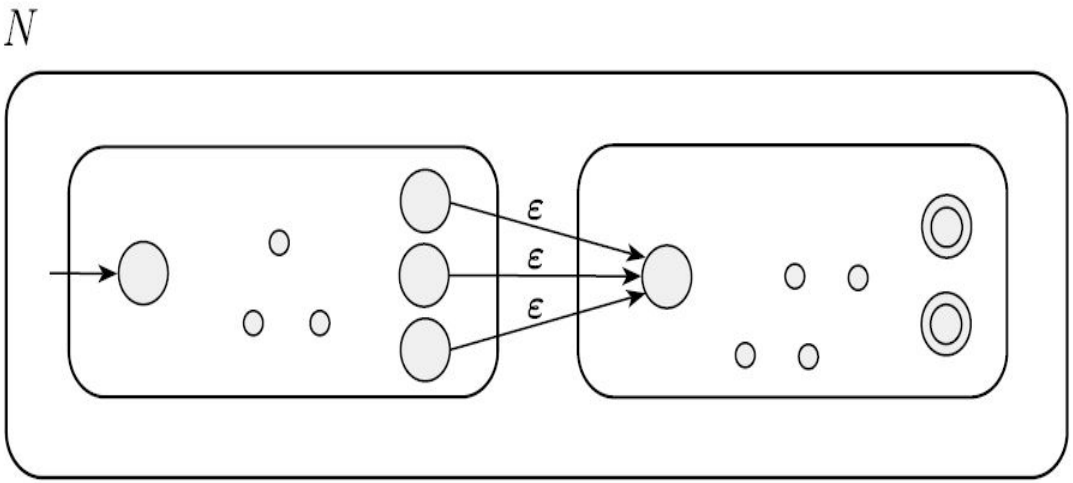
Closure Under Union: Formally

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.
   The states of N are all the states of N_1 and N_2, with the addition of a new start state q_0.
2. The state q_0 is the start state of N.
3. The set of accept states $F = F_1 \cup F_2$.
   The accept states of N are all the accept states of N_1 and N_2. That way, N accepts if either N_1 accepts or N_2 accepts.
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

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

Closure Under Concatenation



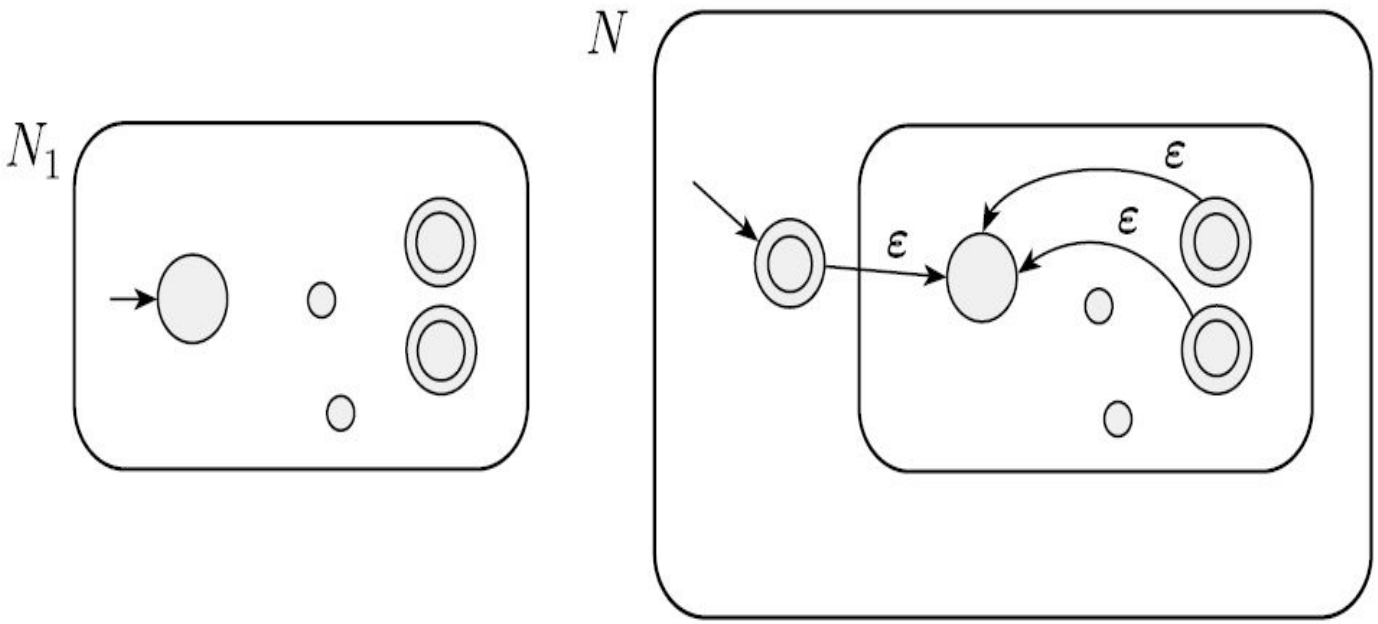


Closure Under Concatenation: Formally

1. $Q = Q_1 \cup Q_2$.
   The states of N are all the states of N_1 and N_2.
2. The state q_1 is the same as the start state of N_1.
3. The accept states F_2 are the same as the accept states of N_2.
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

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

Closure Under Star



Closure Under Star: Formally

1. $Q = \{q_0\} \cup Q_1$.
   The states of N are the states of N_1 plus a new start state.
2. The state q_0 is the new start state.
3. $F = \{q_0\} \cup F_1$.
   The accept states are the old accept states plus the new start state.

4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

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

Regular Expressions (RE)

- Arithmetic operations such as $+$, $\times$ etc.

  are used to build arithmetic expressions.

- Regular operations such as $\cup$, $\circ$, and *

  are used to build regular expressions.

- Arithmetic expressions evaluate to numbers.

- Regular Expressions (RE) evaluate to

  collections of strings, or languages.

This is our next topic.