CS 181 (Introduction to Formal Languages and Automata Theory)

April 17, 2022

1 Deterministic finite automata (DFAs)

1.1 Basic notions

Definition 1.1.1

An **alphabet** is any finite set of symbols.

Example 1.1.2. Binary alphabet: $\{\mathtt{0}, \mathtt{1}\}$

Example 1.1.3. English alphabet: $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{c}\}$

Definition 1.1.4

A **string** is any finite sequence of symbols from a given alphabet.

Example 1.1.5. $\mathtt{001010110101}$

Example 1.1.6. $\mathtt{abracadabra}$

Example 1.1.7. ε (empty string)

Definition 1.1.8

A **language** is a set of strings over a given alphabet.

Example 1.1.9. $\varnothing$ (empty language)

Example 1.1.10. $\{\varepsilon\}$

Example 1.1.11. $\{\mathtt{acclaim}, \mathtt{aim}, \mathtt{brim}, \ldots\}$

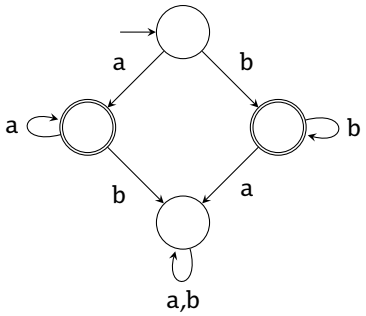
Example 1.1.12. $\{\mathtt{0}, \mathtt{1}, \mathtt{00}, \mathtt{11}, \ldots\}$

Definition 1.1.13

A **computational device** is a mechanism that inputs a string and either accepts or rejects it.

1.2 Deterministic finite automata

- Choose an alphabet: $\{\mathtt{a}, \mathtt{b}\}$.
- Draw states.
- Choose start state and accept states.
- Draw transitions (out of every state on every symbol).



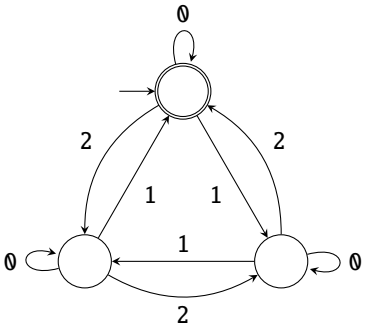
Input	Output
ε	reject
ab	reject
aaa	accept
bb	accept

In words, this machine accepts nonempty strings of all a's or all b's.

Definition 1.2.1

The **language** of a DFA is the set of all strings it accepts.

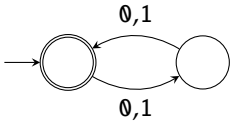
Example 1.2.2.



Input	Output
00...0	accept
12	accept
111	accept
20	reject
1	reject

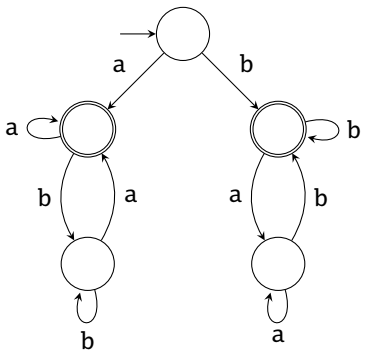
$$\Sigma = \{0, 1, 2\}, \mathcal{L} = \{w : 3 \mid \sum w_i\}$$

Example 1.2.3.



$$\Sigma = \{0, 1\}, \mathcal{L} = \{w : 2 \mid |w|\}$$

Example 1.2.4.

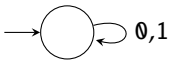


$$\Sigma = \{a, b\}, \mathcal{L} = \left\{w : w \neq \varepsilon \wedge w_1 = w_{|w|}\right\}$$

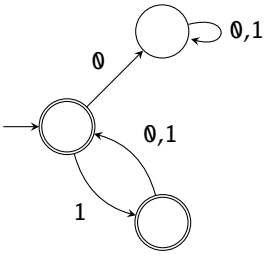
1.3 Designing DFAs

We will be using the binary alphabet $\{0, 1\}$.

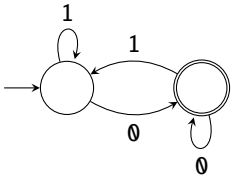
Example 1.3.1. A DFA for $\mathcal{L} = \varnothing$ is given by:



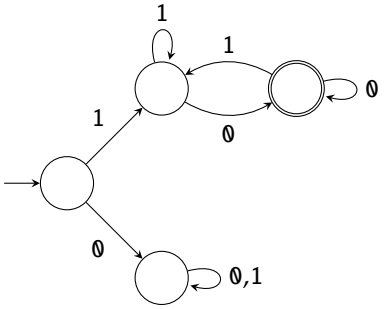
Example 1.3.2. A DFA for $\mathcal{L} = \{w : \text{every odd position is a } 1\}$ is given by:



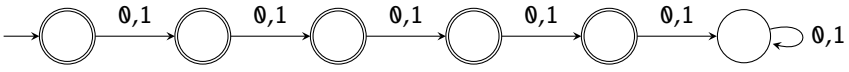
Example 1.3.3. A DFA for $\mathcal{L} = \{w : w \text{ ends in } 0\}$ is given by:



Example 1.3.4. A DFA for $\mathcal{L} = \{w : w \text{ begins with } 1, \text{ ends with } 0\}$ is given by:



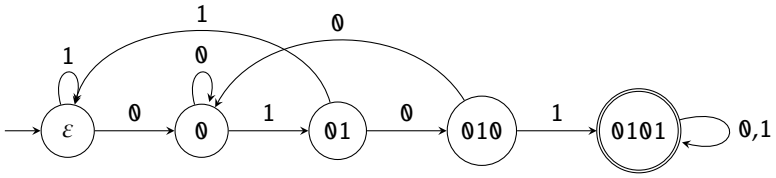
Example 1.3.5. A DFA for $\mathcal{L} = \{w : |w| \leq 4\}$ is given by:



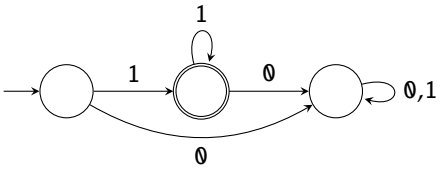
Example 1.3.6. A DFA for $\mathcal{L} = \{w : 1000 \mid |w|\}$ is given by:

In words, each state represents a remainder modulo 1000, and only the 0 state is accepting.

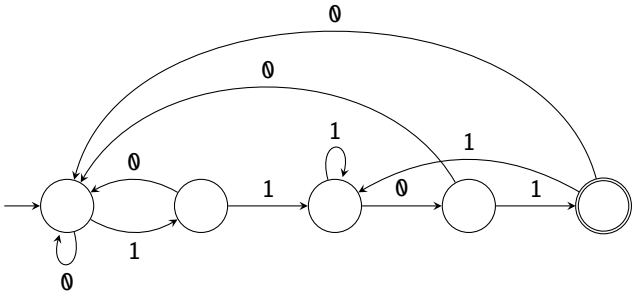
Example 1.3.7. A DFA for $\mathcal{L} = \{w : w \text{ contains } \texttt{0101} \text{ as a substring}\}$ is given by:



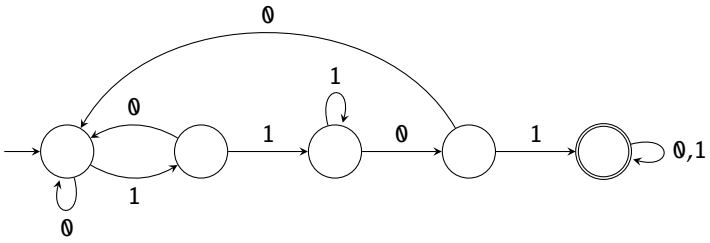
Example 1.3.8 (Week 1 Discussion). A DFA for $\mathcal{L} = \{w : |w| > 0 \wedge w \text{ contains only 1s}\}$ is given by:



Example 1.3.9 (Week 1 Discussion). A DFA for $\mathcal{L} = \{w : w \text{ ends in } \texttt{1101}\}$ is given by:



Example 1.3.10 (Week 1 Discussion). A DFA for $\mathcal{L} = \{w : w \text{ contains } \texttt{1101}\}$ is given by:



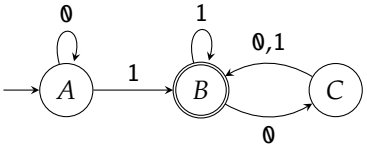
1.4 Formal definitions

Definition 1.4.1

A DFA is a tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q = set of states,
- Σ = alphabet,
- δ = transition function ($\delta \colon Q \times \Sigma \to Q$),
- q_0 = start state ($q_0 \in Q$), and
- F = set of accept states ("favorable"? states, $F \subseteq Q$).

Example 1.4.2. A formal description of the DFA



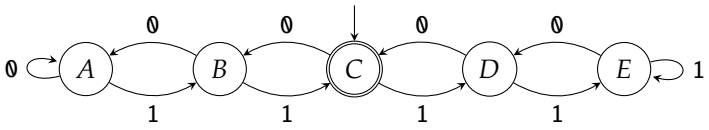
is given by $(\{A, B, C\}, \{0, 1\}, \delta, A, \{B\})$ where δ is defined by the table

	0	1
A	A	B
B	C	B
C	B	B.

Example 1.4.3. The graph for the DFA $(\{A, B, C, D, E\}, \{0, 1\}, \delta, C, \{C\})$ where δ is defined by the table

	0	1
A	A	B
B	A	C
C	B	D
D	C	E
E	D	E

is given by:



Example 1.4.4. A formal description for Example 1.3.6 is given by $(\{0, 1, 2, \ldots, 999\}, \{0, 1\}, \delta, 0, \{0\})$ where $\delta(q, \sigma) = (q + 1) \mod 1000$.

Definition 1.4.5

DFA $(Q, \Sigma, \delta, q_0, F)$ **accepts** a string $w = w_1 w_2 \dots w_n$ iff

$$\delta(\cdots \delta(\delta(q_0, w_1), w_2) \cdots, w_n) \in F.$$

Definition 1.4.6

DFA D **recognizes** the language $\mathcal{L}$ iff

$$\mathcal{L} = \{w : D \text{ accepts } w\}.$$

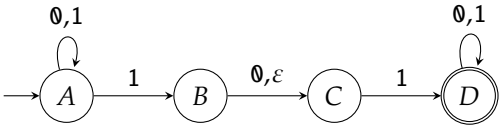
Note.

- Every DFA recognizes exactly 1 language.
- A language has either 0 or ∞ DFAs recognizing it.

2 Nondeterminism

2.1 Basic notions

Example 2.1.1.



- Choose an alphabet: $\{0, 1\}$.
- Draw states.
- Choose start state and accept states. The steps so far are the same as those of a DFA.
- Draw transitions. A state may have any number of transitions on a given symbol. A state may also have transitions on ε.

Definition 2.1.2

An NFA **accepts** w iff there is *at least* one path to an accept state.

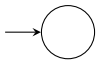
Example 2.1.3. The output table for Example 2.1.1 is given by

Input	Accepting path	Output
ε	-	reject
0	-	reject
1	-	reject
010110	$AABCDDD$	accept
010	-	reject
11	$ABCD$	accept.

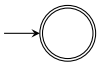
Then the language is the set of all strings containing 101 or 11.

2.2 Using shortcuts

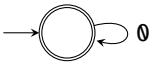
Example 2.2.1. An NFA for $\mathcal{L} = \varnothing$ is given by:



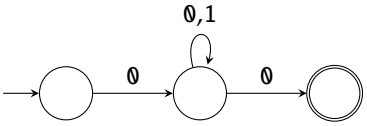
Example 2.2.2. An NFA for $\mathcal{L} = \{\varepsilon\}$ is given by:



Example 2.2.3. An NFA for $\mathcal{L} = \{w : w \text{ doesn't contain } 1\}$ is given by:

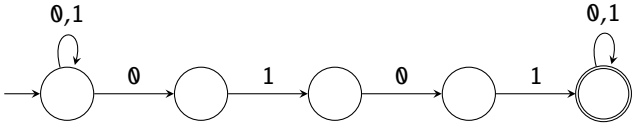


Example 2.2.4. An NFA for $\mathcal{L} = \{w : |w| \geq 2 \text{ and } w \text{ starts and ends with } 0\}$ is given by:

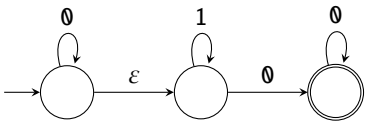


2.3 Pattern matching

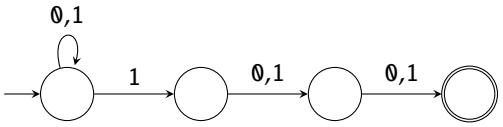
Example 2.3.1. An NFA for $\mathcal{L} = \{w : \text{conatins } 0101\}$ is given by:



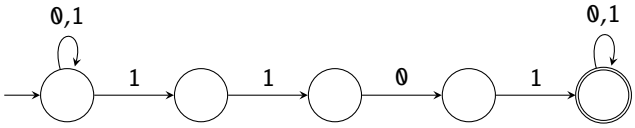
Example 2.3.2. An NFA for $\mathcal{L} = \{w : w = \underbrace{\texttt{00...0}}_{\geq 0}\underbrace{\texttt{11...1}}_{\geq 0}\underbrace{\texttt{00...0}}_{\geq 1}\}$ is given by:



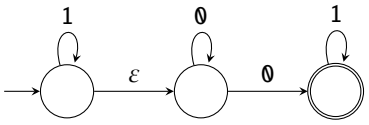
Example 2.3.3. An NFA for $\mathcal{L} = \{w : w \text{ has a 1 in the 3rd position from the end}\}$ is given by:



Example 2.3.4 (Week 1 Discussion). An NFA for $\mathcal{L} = \{w : w \text{ contains } 1101\}$ is given by:

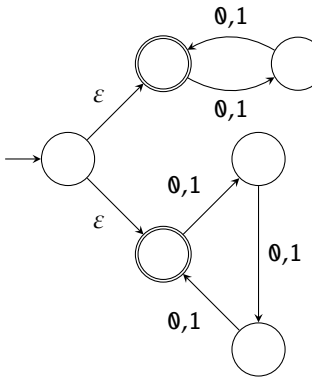


Example 2.3.5 (Week 1 Discussion). An NFA for $\mathcal{L} = \{w : w = \underbrace{\texttt{11...1}}_{\geq 0}\underbrace{\texttt{00...0}}_{\geq 1}\underbrace{\texttt{11...1}}_{\geq 0}\}$ is given by:

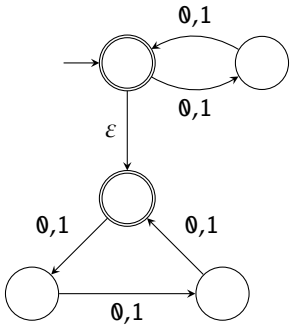


2.4 Alternatives

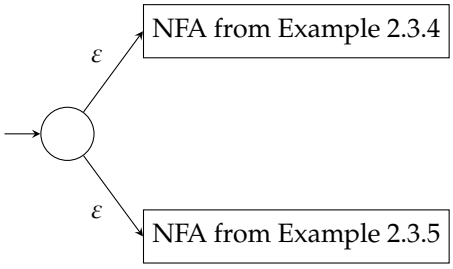
Example 2.4.1. An NFA for $\mathcal{L} = \{w : 2 \mid |w| \vee 3 \mid |w|\}$ is given by:



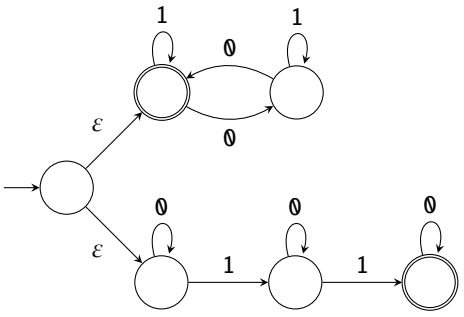
Note that the following is not valid due to side effects:



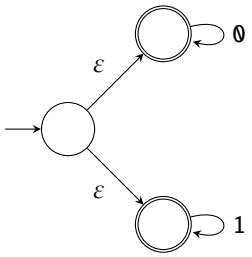
Example 2.4.2 (Week 1 Discussion). An NFA for
$\mathcal{L} = \{w : w \text{ contains } 1101 \vee w = \underbrace{11\ldots1}_{\geq 0}\underbrace{00\ldots0}_{\geq 1}\underbrace{11\ldots1}_{\geq 0}\}$ is given by:



Example 2.4.3. An NFA for $\mathcal{L} = \{w : w \text{ contains an even number of } 0\text{s, or exactly two } 1\text{s}\}$ is given by:



Example 2.4.4. An NFA for $\mathcal{L} = \{w : w \text{ does not contain both } 0 \text{ and } 1\}$ is given by:



2.5 Formal definitions

Definition 2.5.1

An NFA is a tuple $(Q, \Sigma, \delta q_0, F)$ where

- Q = set of states,
- Σ = alphabet,
- δ = transition function ($\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$),
- q_0 = start state ($q_0 \in Q$), and
- F = set of accept states ($F \subseteq Q$).

Example 2.5.2. A formal description for NFA from Example 2.1.1 is given by

$$(\{A, B, C, D\}, \{\mathtt{0}, \mathtt{1}\}, \delta, A, \{D\})$$

where δ is defined by the table

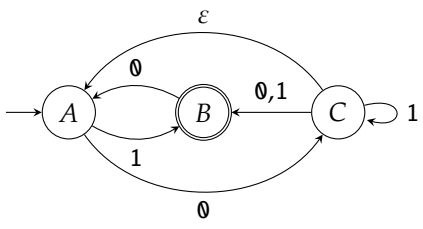
	$\mathtt{0}$	$\mathtt{1}$	ε
A	$\{A\}$	$\{A, B\}$	$\varnothing$
B	$\{C\}$	$\varnothing$	$\{C\}$
C	$\varnothing$	$\{D\}$	$\varnothing$
D	$\{D\}$	$\{D\}$	$\varnothing$.

Note that as pictures are not precise, we had to make an assumption on the alphabet of the NFA. Also note that although adding transition from A to A on ε does not change the language, it does not match the drawing and therefore represents a different NFA. Make sure to transcribe the given NFA.

Example 2.5.3. The graph for the DFA $(\{A, B, C\}, \{\mathtt{0}, \mathtt{1}\}, \delta, A, \{B\})$ where δ is defined by the table

	$\mathtt{0}$	$\mathtt{1}$	ε
A	$\{C\}$	$\{B\}$	$\varnothing$
B	$\{A\}$	$\varnothing$	$\varnothing$
C	$\{B\}$	$\{B, C\}$	$\{A\}$

is given by



Definition 2.5.4

NFA $(Q, \Sigma, \delta, q_0, F)$ **accepts** a string w iff

$$\begin{aligned}\exists\, q_1, q_2, \ldots, q_m \in Q\, \exists\, \sigma_0, \sigma_1, \ldots, \sigma_{m-1} \in \Sigma \cup \{\varepsilon\} :& \\ q_1 \in \delta(q_0, \delta_0) \wedge q_2 \in \delta(q_1, \sigma) \wedge \cdots& \\ \wedge\, q_m \in \delta(q_{m-1}, \sigma_{m-1}) \wedge q_m \in F& \\ \wedge\, \sigma_0\sigma_1 \ldots \sigma_{m-1} = w,&\end{aligned}$$

or, in words, an accept state is reachable from q_0 via some path on input w.

Definition 2.5.5

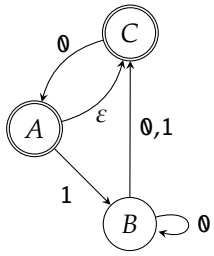
NFA N **recognizes** the language $\mathcal{L}$ iff

$$\mathcal{L} = \left\{w : N \text{ accepts } w\right\}.$$

3 Equivalence of DFAs and NFAs

3.1 Example

Does this NFA accept the string `0110`?



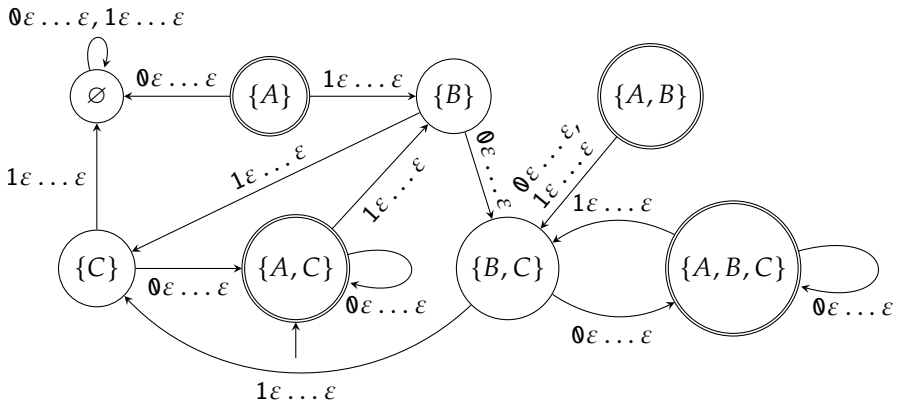
Recall that the NFA accepts `0110` iff an accept state is reachable from the start state via some path $\underbrace{\varepsilon \ldots \varepsilon}_{\geq 0}\,\mathtt{0}\,\underbrace{\varepsilon \ldots \varepsilon}_{\geq 0}\,\mathtt{1}\,\underbrace{\varepsilon \ldots \varepsilon}_{\geq 0}\,\mathtt{1}\,\underbrace{\varepsilon \ldots \varepsilon}_{\geq 0}\,\mathtt{0}\,\underbrace{\varepsilon \ldots \varepsilon}_{\geq 0}$.

Path	Possible end states
$\varepsilon \ldots \varepsilon$	A, C
$\varepsilon \ldots \varepsilon \mathtt{0} \varepsilon \ldots \varepsilon$	A, C
$\varepsilon \ldots \varepsilon \mathtt{0} \varepsilon \ldots \varepsilon \mathtt{1} \varepsilon \ldots \varepsilon$	B
$\varepsilon \ldots \varepsilon \mathtt{0} \varepsilon \ldots \varepsilon \mathtt{1} \varepsilon \ldots \varepsilon \mathtt{1} \varepsilon \ldots \varepsilon$	C
$\varepsilon \ldots \varepsilon \mathtt{0} \varepsilon \ldots \varepsilon \mathtt{1} \varepsilon \ldots \varepsilon \mathtt{1} \varepsilon \ldots \varepsilon \mathtt{0} \varepsilon \ldots \varepsilon$	A, C

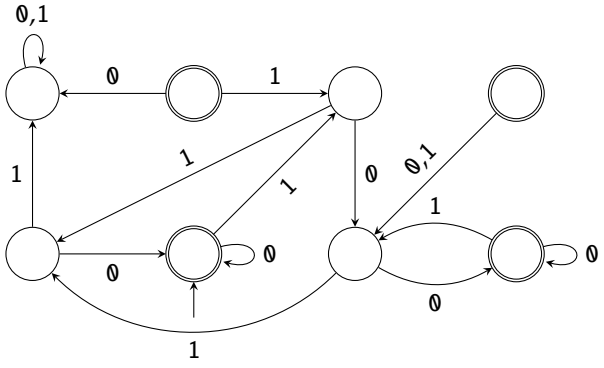
Since C is an accept state, the NFA accepts the string `0110`.

Can we convert this to a DFA? Yes.

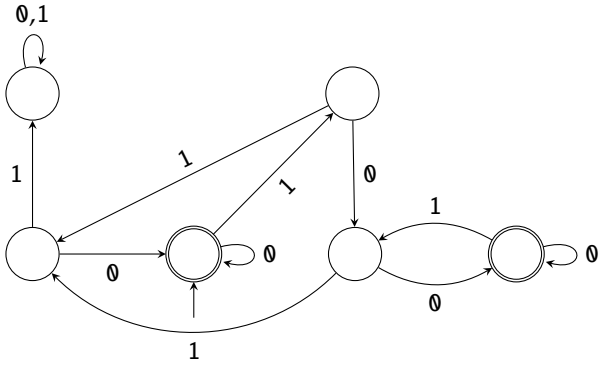
Step 1 Use subsets of the states of the NFA as the states of the DFA. Accept states are subsets that contain accept states of the NFA. The start state is the subset of all reachable states from the start state of the NFA via ε. For subsets containing more than one element, the transition is the union of all transitions for each individual element.



Step 2 Clean up.



Step 3 Optionally drop unreachable states.



3.2 General theorem

Theorem 3.2.1

Every NFA N can be converted to a DFA D that recognizes the same language.

Proof. Given NFA $N = (Q, \Sigma, \delta, q_0, F)$ define DFA $D = (\mathcal{P}(Q), \Sigma, \Delta, S_0, \mathcal{F})$ where

- $S_0 = \{q \in Q : q \text{ is reachable from } q_0 \text{ via a path } \underbrace{\varepsilon \ldots \varepsilon}_{\geq 0}\}$,
- $\Delta(S, \sigma) = \{q \in Q : q \text{ is reachable from a state in } S \text{ via a path } \delta \underbrace{\varepsilon \ldots \varepsilon}_{\geq 0}\}$, and
- $\mathcal{F} = \{S \subseteq Q : S \text{ contains a state in } F\}$.

Then N accepts a string $w = w_1 w_2 \ldots w_n \Leftrightarrow$ a state in F is reachable via a path $\underbrace{\varepsilon \ldots \varepsilon}_{\geq 0} w_1 \underbrace{\varepsilon \ldots \varepsilon}_{\geq 0} w_2 \ldots w_n \underbrace{\varepsilon \ldots \varepsilon}_{\geq 0}$
$\Leftrightarrow$ a state in $\mathcal{F}$ is reachable from S_0 via the path $w_1 w_2 \ldots w_n \Leftrightarrow D$ accepts w. □

3.3 Blow-up in size

Definition 3.3.1

A language $\mathcal{L}$ is called **regular** iff it is recognized by some NFA or, equivalently, some DFA.

Last time: If $\mathcal{L}$ has an NFA with k states then $\mathcal{L}$ has a DFA with 2^k states.

We will show that this conclusion is the best possible.

Theorem 3.3.2

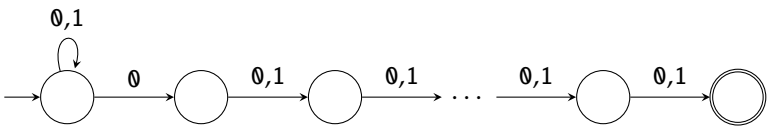
Define $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, k = any positive integer, $\mathcal{L}_k = \{w : w \text{ has a } \mathtt{0} \text{ in the } k\text{-th position from the end}\}$.

(a) $\mathcal{L}_k$ has a NFA of size $k + 1$, and

(b) $\mathcal{L}_k$ does not have a DFA of size $< 2^k$.

Proof.

(a)



(b) Let D be any DFA with $< 2^k$ states.

Input	End state of D
`000...00`	q_i
`000...01`	q_j
$\vdots$	$\vdots$
`111...11`	q_k

If the left hand side of the table enumerates all 2^k strings of length k, then the right hand side cannot be all distinct due to the pigeonhole principle. Then there exists strings $u \neq v$ of length k such that

$$\text{end state on } u = \text{end state on } v.$$

Say $u_i = 0$, $v_i = 1$. Then

$$\text{end state on } u\underbrace{\texttt{00...0}}_{i-1} = \text{end state on } v\underbrace{\texttt{00...0}}_{i-1}.$$

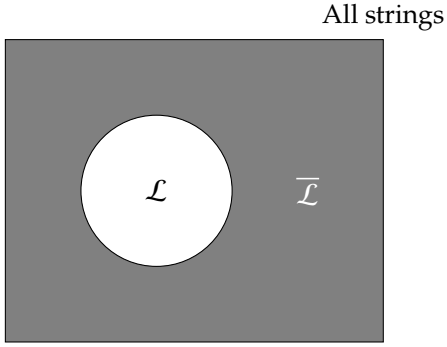
However, note that the left hand side has a `0` in the k-th position from the end while the right hand side has a `1` in the k-th position from the end.

Then we conclude that D does not recognize $\mathcal{L}_k$.

□

4 Closure

4.1 Complement



Definition 4.1.1

The **complement** of the language $\mathcal{L}$ is

$$\overline{\mathcal{L}} = \{w : w \notin \mathcal{L}\}.$$

Theorem 4.1.2

If $\mathcal{L}$ is regular then $\overline{\mathcal{L}}$ is regular. In other words, regular languages are closed under complement.

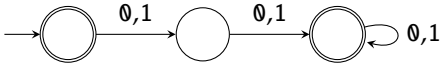
Proof. Swap the accept and reject states of the DFA for $\mathcal{L}$ to get a DFA for $\overline{\mathcal{L}}$.

Formally, if DFA $(Q, \Sigma, \delta, q_0, F)$ recognizes $\mathcal{L}$ then DFA $(Q, \Sigma, \delta, q_0, Q \setminus F)$ recognizes $\overline{\mathcal{L}}$. □

Example 4.1.3. $\mathcal{L} = \{w : w \neq \mathtt{0}, \mathtt{1}\}$.

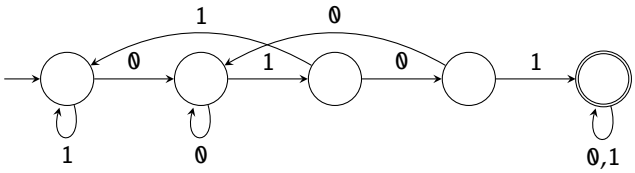
Note that $\overline{\mathcal{L}} = \{w : w = \mathtt{0} \vee w = \mathtt{1}\}$. The DFA for $\overline{\mathcal{L}}$ is given by:

Then the DFA for $\mathcal{L}$ is given by

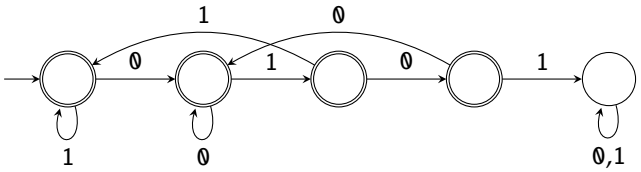


Example 4.1.4. $\mathcal{L} = \{w : w \text{ does not contain } \mathtt{0101}\}$.

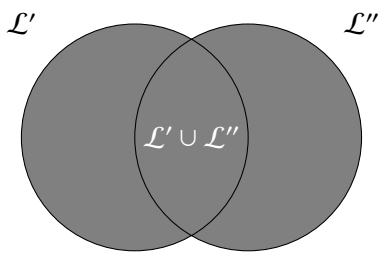
Note that $\overline{\mathcal{L}} = \{w : w \text{ contains } \mathtt{0101}\}$. The DFA for $\overline{\mathcal{L}}$ is given by:

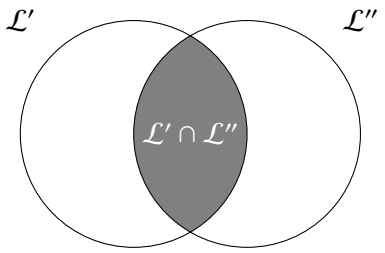


Then the DFA for $\mathcal{L}$ is given by



4.2 Union and intersection





Definition 4.2.1

The **union** of languages $\mathcal{L}'$ and $\mathcal{L}''$ is

$$\mathcal{L}' \cup \mathcal{L}'' = \{w : w \in \mathcal{L}' \vee w \in \mathcal{L}''\}.$$

Definition 4.2.2

The **intersection** of languages $\mathcal{L}'$ and $\mathcal{L}''$ is

$$\mathcal{L}' \cap \mathcal{L}'' = \{w : w \in \mathcal{L}' \wedge w \in \mathcal{L}''\}.$$

Theorem 4.2.3

If $\mathcal{L}'$, $\mathcal{L}''$ are regular, then so are

(a) $\mathcal{L}' \cup \mathcal{L}''$ and

(b) $\mathcal{L}' \cap \mathcal{L}''$.

In other words, regular lanaguages are closed under union and intersection.

Proof.

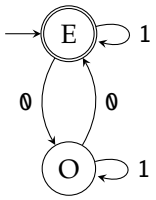
(a) Connect an initial state to the initial states for the NFAs for $\mathcal{L}'$ and $\mathcal{L}''$ via ε.

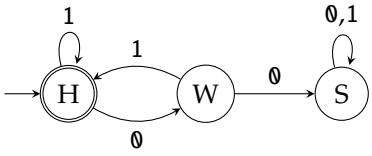
(b) If $\mathcal{L}'$ and $\mathcal{L}''$ are regular then $\overline{\mathcal{L}'}$ and $\overline{\mathcal{L}''}$ are regular. Then $\overline{\mathcal{L}'} \cup \overline{\mathcal{L}''}$ is regular. Then, by De Morgan's law, $\mathcal{L}' \cap \mathcal{L}'' = \overline{\overline{\mathcal{L}'} \cup \overline{\mathcal{L}''}}$ is regular.

□

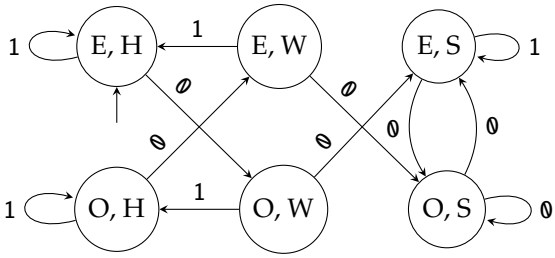
Proof. **Idea:** [1]

$\mathcal{L}' = \{w : w$ has an even number of $\mathtt{0}$s$\}$. $\mathcal{L}'' = \{w :$ every $\mathtt{0}$ is immediately followed by a $\mathtt{1}\}$.





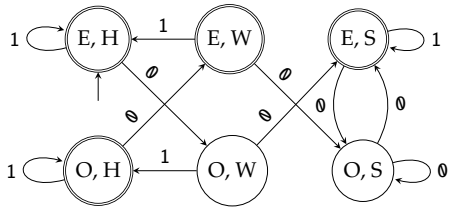
Consider the construction:



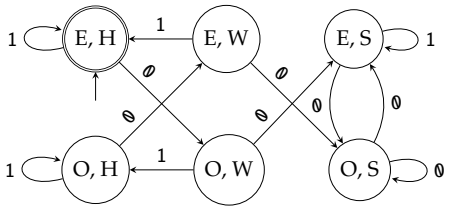
[1] Here "E" stands for "EVEN", "O" stands for "ODD", "H" stands for "HAPPY", "W" stands for "WORRIED", and "S" stands for "SAD".

Note that we can create DFAs for either union or intersection by assigning appropriate accept states:

$\mathcal{L} = \mathcal{L}' \cup \mathcal{L}''$.



$\mathcal{L} = \mathcal{L}' \cap \mathcal{L}''$.



Note that this construction is better than that in the first proof since the number of nodes in this construction is $O(n^2)$ while that in the first proof is $O(4^n)$.

Formally, write the DFA for $\mathcal{L}'$ as $(Q', \Sigma, \delta', q_0', F')$ and that for $\mathcal{L}''$ as $(Q'', \Sigma, \delta'', q_0'', F'')$.

Then the DFA for $\mathcal{L}' \cup \mathcal{L}''$ is given by $(Q' \times Q'', \Sigma, \delta, (q_0', q_0''), (F' \times Q'') \cup (Q' \times F''))$ where $\delta((q', q''), \sigma) := (\delta'(q', \sigma), \delta''(q'', \delta))$ for all $q' \in Q'$, $q'' \in Q''$, and $\sigma \in \Sigma$.

Similarly, the DFA for $\mathcal{L}' \cap \mathcal{L}''$ is given by $(Q' \times Q'', \Sigma, \delta, (q_0', q_0''), F' \times F'')$ where $\delta((q', q''), \sigma) := (\delta'(q', \sigma), \delta''(q'', \delta))$ for all $q' \in Q'$, $q'' \in Q''$, and $\sigma \in \Sigma$. □

Corollary 4.2.4

If $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$ are regular, so are

- $\mathcal{L}_1 \cup \mathcal{L}_2 \cup \cdots \cup \mathcal{L}_n$ and
- $\mathcal{L}_1 \cap \mathcal{L}_2 \cap \cdots \cap \mathcal{L}_n$.

Proof. Induction. □

Corollary 4.2.5

Every finite language is regular.

Proof. Let $\mathcal{L}$ be finite. Then we can write

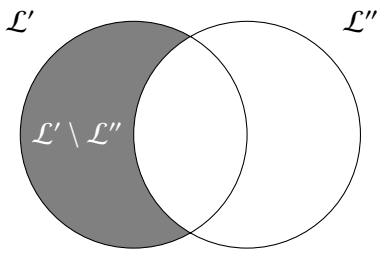
$$\mathcal{L} = \bigcup_{w \in \mathcal{L}} \{w\},$$

that is, as a union of singleton languages.

Note that an NFA can be easily found for any singleton language, that is, singleton languages are regular. Then $\mathcal{L}$ is regular by Corollary 4.2.4. □

4.3 Difference

$$\mathcal{L}' \setminus \mathcal{L}'' = \{w \in \mathcal{L}' : w \notin \mathcal{L}''\}$$



Theorem 4.3.1

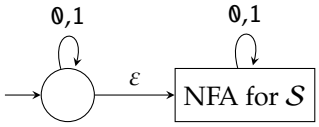
If $\mathcal{L}'$ and $\mathcal{L}''$ are regular then so is $\mathcal{L}' \setminus \mathcal{L}''$.

Proof. $\mathcal{L}' \setminus \mathcal{L}'' = \mathcal{L}' \cap \overline{\mathcal{L}''}$. □

Corollary 4.3.2

If $\mathcal{A}, \mathcal{S}$ are regular, then so is $\mathcal{L} = \{w \in \mathcal{A} : \text{no substring of } w \text{ is in } \mathcal{S}\}$.

Proof. Let $\mathcal{S}' = \{w : \text{some substring of } w \text{ is in } \mathcal{S}\}$. Note that $\mathcal{S}'$ is a superset of $\mathcal{S}$. An NFA for $\mathcal{S}'$ is given by:



where the self loop on NFA for $\mathcal{S}$ refers to self loops on the accept states of the NFA.

Since $\mathcal{S}'$ is regular by construction, $\mathcal{L} = \mathcal{A} \setminus \mathcal{S}'$ is regular. □

4.4 Concatenation

Definition 4.4.1

The **concatenation** of two languages $\mathcal{L}'$ and $\mathcal{L}''$, is given by

$$\mathcal{L}'\mathcal{L}'' = \{w'w'' : w' \in \mathcal{L}' \quad \wedge \quad w'' \in \mathcal{L}''\}.$$

This is sometimes denoted as $\mathcal{L}' \circ \mathcal{L}''$.

Example 4.4.2. Given $\mathcal{L}' = \{000, 111\}$ and $\mathcal{L}'' = \{10, 010\}$, we have

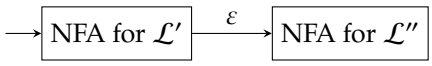
$$\mathcal{L}'\mathcal{L}'' = \{00010, 000010, 11110, 111010\}.$$

Observation: $\mathcal{L}\varnothing\varnothing\mathcal{L} = \varnothing$ and $\mathcal{L}\{\varepsilon\} = \{\varepsilon\}\mathcal{L} = \mathcal{L}$ for any language $\mathcal{L}$.

Theorem 4.4.3

If $\mathcal{L}'$ and $\mathcal{L}''$ are regular then so is $\mathcal{L}'\mathcal{L}''$.

Proof. An NFA for $\mathcal{L}'\mathcal{L}''$ is given by:



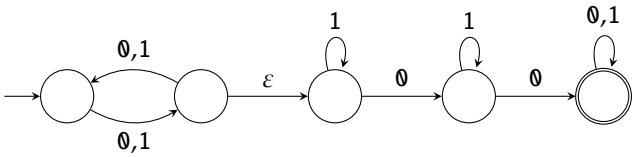
where the ε transition from the first NFA to the second NFA refer to those from the accept states in the first NFA to the start state of the second and all accept states in the first NFA are converted to reject states.

Formally, write the first NFA as $(Q', \Sigma, \delta', q_0', F')$ and the second NFA as $(Q'', \Sigma, \delta', q_0'', F'')$. Then we

- make every state in F' a reject state and
- add ε transitions from every state in F' to q_0''.

□

Example 4.4.4. An NFA for $\{uw : |u| \text{ is odd, } v \text{ contains at least two } \mathtt{0}\text{s}\}$ is given by:



4.5 Kleene star

Definition 4.5.1

The **Kleene star** of a language $\mathcal{L}$ is given by

$$\mathcal{L}^* = \left\{w : w \text{ is the concatenation of zero or more strings in } \mathcal{L}\right\}.$$

Example 4.5.2. The Kleene star of $\mathcal{L} = \{\mathtt{a}, \mathtt{b}\}$ is given by

$$\mathcal{L}^* = \{\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}, \dots\}.$$

In words, $\{\mathtt{a}, \mathtt{b}\}^*$ = the set of all strings over the alphabet $\{\mathtt{a}, \mathtt{b}\}$.

Observation: $\varnothing^* = \{\varepsilon\}^* = \{\varepsilon\}$.

Observation: $\mathcal{L}^*$ is finite if and only if $\mathcal{L} = \varnothing$ or $\mathcal{L} = \{\varepsilon\}$.

Theorem 4.5.3

If $\mathcal{L}$ is regular then so is $\mathcal{L}^*$.

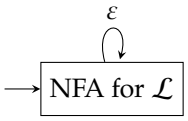
Proof. (Incorrect) Note that

$$\mathcal{L}^* = \{\varepsilon\} \cup \mathcal{L} \cup \mathcal{L}\mathcal{L} \cup \cdots \cup \underbrace{\mathcal{L} \cdots \mathcal{L}}_{n} \cup \cdots,$$

and since the union of regular languages is regular, $\mathcal{L}^*$ is regular.

However, note that the theorem that the union of regular languages is regular does not work for an infinite number of languages and therefore this proof fails. □

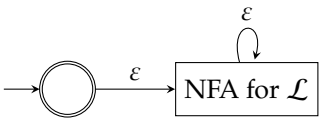
Proof. (Incorrect) An NFA for $\mathcal{L}^*$ is given by



where the self loop on the NFA refers to ε transitions from the accept states of the NFA to the start state and the start state is converted to an accept state.

However, this construction fails if q_0 is originally rejecting *and* part of a cycle. □

Proof. (Correct) An NFA for $\mathcal{L}^*$ is given by



where the self loop on the NFA refers to ε transitions from the accept states of the NFA to the start state.

In words, add ε-transitions from every state in F to q_0 and add a new start state q_0' and link it with an ε-arrow to q_0. □

4.6 Reverse

Definition 4.6.1

For a string $w = w_1 w_2 \dots w_n$, its **reverse** is given by

$$w^R = w_n w_{n-1} \dots w_2 w_1.$$

Definition 4.6.2

The **reverse** of a language $\mathcal{L}$ is given by

$$\mathcal{L}^R = \left\{ w^R : w \in \mathcal{L} \right\}.$$

Example 4.6.3. $\{\texttt{011}, \texttt{abc}\}^R = \{\texttt{110}, \texttt{cba}\}$.

Theorem 4.6.4

If $\mathcal{L}$ is regular then so is $\mathcal{L}^R$.

Proof. Consider an NFA $(Q, \Sigma, \delta, q_0, F)$ for $\mathcal{L}$. Reverse all the transitions, create a new start state, and link it with ε-transitions to states in F. Make q_0 the only accept state. □

4.7 Prefix and suffix

Definition 4.7.1

The **prefix language** of a language $\mathcal{L}$ is given by

$$\text{prefix}(\mathcal{L}) = \left\{u : uv \in \mathcal{L} \text{ for some string } v, \text{possibly empty}\right\}.$$

Definition 4.7.2

The **suffix language** of a language $\mathcal{L}$ is given by

$$\text{suffix}(\mathcal{L}) = \left\{v : uv \in \mathcal{L} \text{ for some string } u, \text{possibly empty}\right\}.$$

Theorem 4.7.3

If $\mathcal{L}$ is regular then so is prefix($\mathcal{L}$).

Proof. Given a DFA $(Q, \Sigma, \delta, q_0, F)$ for $\mathcal{L}$, a DFA for prefix($\mathcal{L}$) is given by $(Q, \Sigma, \delta, q_0, F')$ where

$$F' = \left\{q \in Q : \text{there exists a path, possibly empty, from } q \text{ to a state in } F\right\}$$

□

Corollary 4.7.4

If $\mathcal{L}$ is regular then so is suffix($\mathcal{L}$).

Proof. $\text{suffix}(\mathcal{L}) = (\text{prefix}(\mathcal{L}^R))^R$. □

4.8 Substring

Definition 4.8.1

The **substring language** of a language $\mathcal{L}$ is given by

$$\text{substring}(\mathcal{L}) = \left\{v : uvw \in \mathcal{L} \text{ for some strings } u, w\right\}.$$

Theorem 4.8.2

If $\mathcal{L}$ is regular then so is substring($\mathcal{L}$).

Proof. $\text{substring}(\mathcal{L}) = \text{suffix}(\text{prefix}(\mathcal{L}))$. □

4.9 Shuffle

Definition 4.9.1

The **shuffle language** of two languages $\mathcal{L}'$ and $\mathcal{L}''$ is given by

$$\left\{a_1b_1a_2b_2\dots a_kb_k : a_1,\dots,a_k \text{ strings}, b_1,\dots,b_k \text{ strings}, a_1\cdots a_k \in \mathcal{L}', b_1\cdots b_k \in \mathcal{L}''\right\}.$$

Proof. Given an NFA $(Q', \Sigma, \delta', q_0', F')$ for $\mathcal{L}'$ and an NFA $(Q'', \Sigma, \delta'', q_0'', F'')$ for $\mathcal{L}''$, an NFA for $\mathcal{L}' \diamond \mathcal{L}''$ is given by $(Q' \times Q'', \Sigma, \Delta, (q_0', q_0''), F' \times F'')$ where

$$\Delta((q', q''), \sigma) = \left(\delta'(q', \sigma) \times \left\{q''\right\}\right) \cup \left(\left\{q'\right\} \times \delta''(q'', \sigma)\right).$$

□

5 Regular expressions

5.1 Basic notions

Definition 5.1.1

A regular expression is

- $\varnothing$,
- $\{\varepsilon\}$,
- $\{\sigma\}$ for $\sigma \in \Sigma$,
- (R^*),
- $(R_1 \cup R_2)$, and
- $(R_1 \circ R_2)$.

Example 5.1.2. $((((\{\mathtt{0}\} \cup \{\mathtt{1}\})^*) \cup \{\mathtt{1}\})$.

Example 5.1.3. $((((\{\mathtt{0}\} \circ \{\mathtt{1}\}) \circ \{\mathtt{1}\}) \circ \{\mathtt{1}\})$.

Example 5.1.4. $(((((\{\mathtt{0}\} \circ \{\mathtt{1}\})^*) \circ \{\mathtt{1}\}) \circ ((\{\mathtt{0}\} \cup \{\mathtt{1}\})^*))$

Shorthand

- $R^k = \underbrace{(R \circ R \circ \cdots \circ R)}_{k}$
- $R^+ = (R \circ (R^*))$
- $\Sigma = (\{\mathtt{0}\} \cup \{\mathtt{1}\})$ assuming alpha bet is $\{\mathtt{0}, \mathtt{1}\}$
- may omit $\{\}$, $()$, $\circ$

Precedence: $*$ then $\circ$ then $\cup$.

Note. Sometimes () are necessary.

Example 5.1.5. Example 5.1.2 can be simplified to $\Sigma^* \cup \{1\} = \Sigma^*$.

Example 5.1.6. Example 5.1.3 can be simplified to $\mathtt{0}111$.

Example 5.1.7. Example 5.1.4 can be simplified to $\Sigma^*1\Sigma^*$.

Example 5.1.8. A regular expression for $\{w : w \text{ contains exactly one } 1\}$ is $\mathtt{0}^*1\mathtt{0}^*$.

Example 5.1.9. A regular expression for $\{w : w \text{ contains at least one } 1\}$ is $\Sigma^*1\Sigma^*$.

Example 5.1.10. A regular expression for $\{w : w \text{ contains } \mathtt{00}1 \text{ as a substring}\}$ is $\Sigma^*\mathtt{00}1\Sigma^*$.

Example 5.1.11. A regular expression for $\{w : w \text{ does not contain } \mathtt{00} \text{ as a substring}\}$ is $1^*(\mathtt{0}1^+)^*(\varepsilon \cup \mathtt{0})$.

Example 5.1.12. A regular expression for $\{w : |w| \text{ is even}\}$ is $(\Sigma^2)^*$.

Example 5.1.13. A regular expression for $\{w : 5 \mid |w|\}$ is $(\Sigma^5)^*$.

Example 5.1.14. A regular expression for $\{w : w \neq \varepsilon \wedge w_1 = w_{|w|}\}$ is $\Sigma \cup \mathtt{0}\Sigma^*\mathtt{0} \cup 1\Sigma^*1$.

Example 5.1.15. A regular expression for $\{\mathtt{01}, \mathtt{10}\}$ is $\mathtt{01} \cup \mathtt{10}$.

Example 5.1.16. The regular expression $(\mathtt{0} \cup \varepsilon)1^*$ can be arguably simplified to $\mathtt{0}1^* \cup 1^*$.

Example 5.1.17. The regular expression $1^*\varnothing$ can be simplified to $\varnothing$.

Example 5.1.18. The regular expression $\varnothing^*$ can be simplified to ε.

Example 5.1.19. The regular expression $(\mathtt{0} \cup 1^+\mathtt{0})^*$ can be simplified to $((\varepsilon \cup 1^+)\mathtt{0})*$ and then to $(1^*\mathtt{0})^*$. In words, all strings that do not end in a 1.

Example 5.1.20.

Regular expression	Shortest accepted string	Shortest rejected string
$\mathtt{0}^*\mathtt{00}1^*$	$\mathtt{00}1$	ε
$\mathtt{0}^*\mathtt{00}(\mathtt{0} \cup 1)^*$	$\mathtt{00}$	ε
$(\mathtt{00})^*(11\mathtt{0})^*(11)^*$	ε	$\mathtt{0}$, 1
$\mathtt{0}^*(\mathtt{0}1 \cup 1\mathtt{0})^*1^*$	ε	$1\mathtt{00}$, $11\mathtt{0}$

5.2 Equivalence with automata

Theorem 5.2.1

The language of every regular expression is regular.

Proof. Induction. □

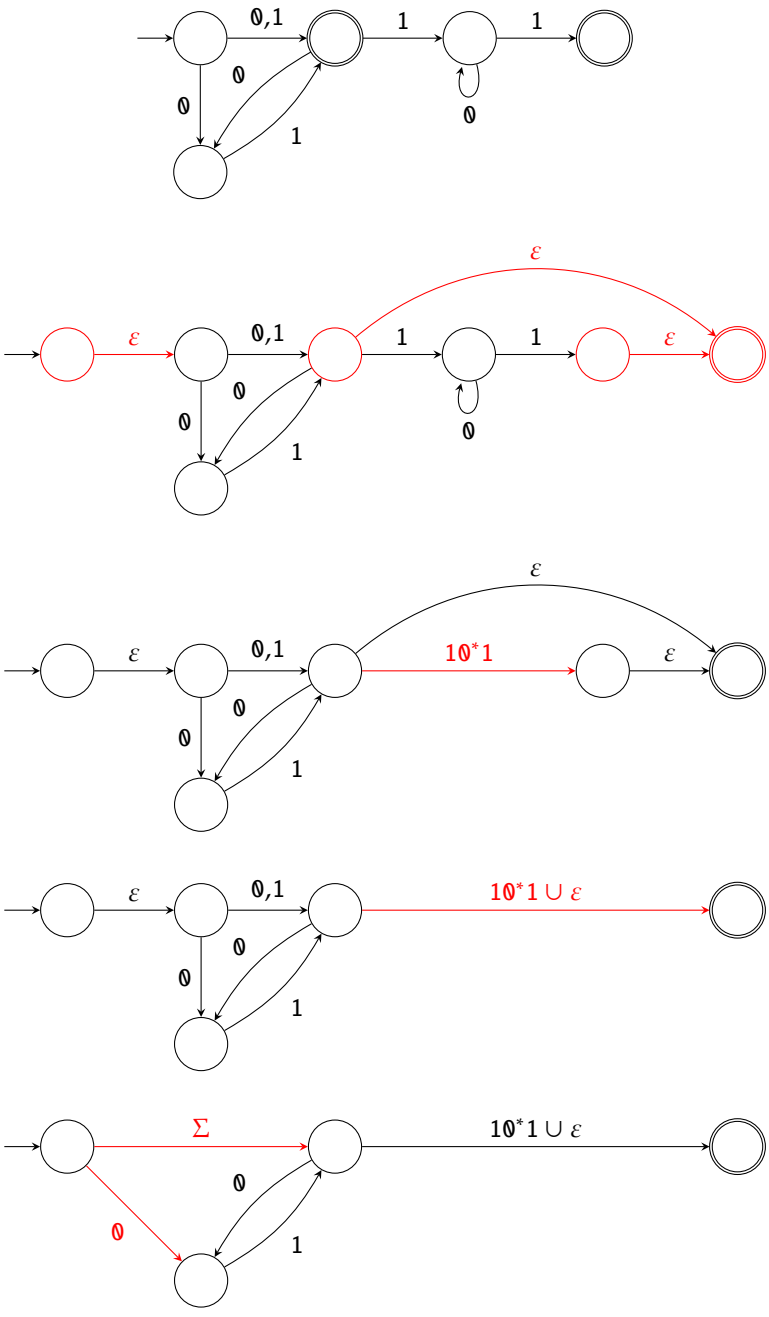
Theorem 5.2.2: Kleene's theorem

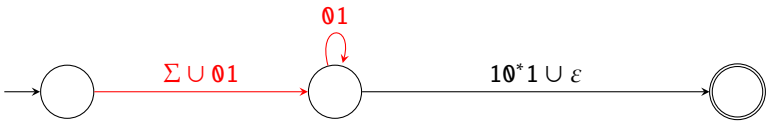
Then language of every NFA can be represented by a regular expression.

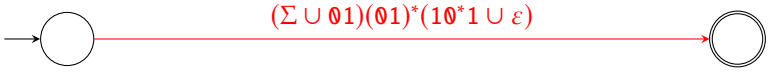
Proof. Given an NFA, create a new state state and a new and unique accept state (making every state in the original NFA rejecting). Choose a state other than the start state and the accept state. Call the states from which the incoming transitions originate donors and those at which the outgoing transitions arrive receivers. Note that a state can be both a donor and a receiver. Then for each donor-receiver pair, replace the transitions with a single transition $\alpha\beta^*\gamma$ where α is the incoming transition, β is the looping transition, and γ

is the outgoing transition. Remove the chosen state. Collapse any parallel edges created in the process by unioning them. If there are states other than the start state or the accept state, repeat the process. If there are no transitions from the start state to the accept state then the regular expression is $\varnothing$. Otherwise there is exactly one transition α, the desired regular expression. □

5.3 Conversion example




01
Σ ∪ 01
10*1 ∪ ε


(Σ ∪ 01)(01)*(10*1 ∪ ε)