CS 181 (Introduction to Formal Languages and Automata Theory)

August 7, 2022

Contents

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: $\{\texttt{0}, \texttt{1}\}$

Example 1.1.3. English alphabet: $\{\texttt{a}, \texttt{b}, \dots, \texttt{c}\}$

Definition 1.1.4

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

Example 1.1.5. `001010110101`

Example 1.1.6. `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. $\{\texttt{acclaim}, \texttt{aim}, \texttt{brim}, \dots\}$

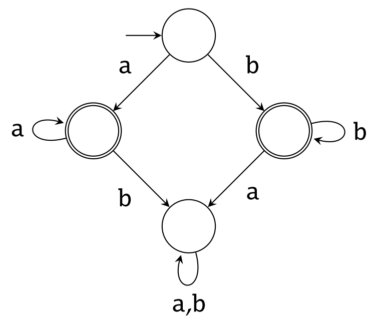
Example 1.1.12. $\{\texttt{0}, \texttt{1}, \texttt{00}, \texttt{11}, \dots\}$

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).



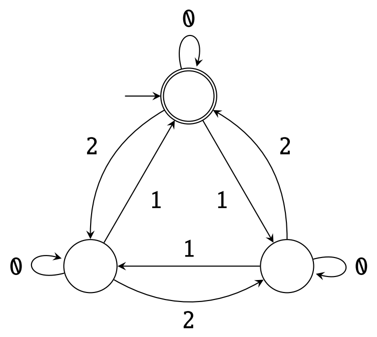
Input	Output
ε	reject
ab	reject
aaa	accept
bb	accept

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.

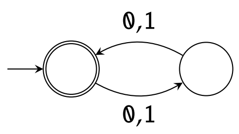
Example 1.2.2.



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

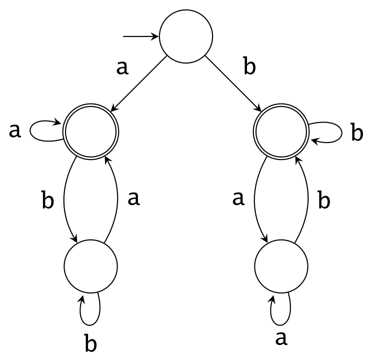
$$\Sigma = \{\mathtt{0}, \mathtt{1}, \mathtt{2}\},\ \mathcal{L} = \{w : 3 \mid \textstyle\sum w_i\}$$

Example 1.2.3.



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

Example 1.2.4.

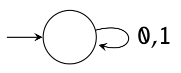


$$\Sigma = \{\mathtt{a}, \mathtt{b}\},\ \mathcal{L} = \left\{w : w \neq \varepsilon \wedge w_1 = w_{|w|}\right\}$$

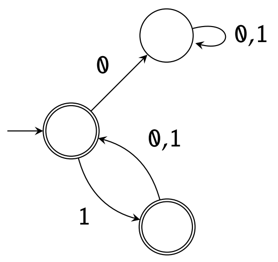
1.3 Designing DFAs

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

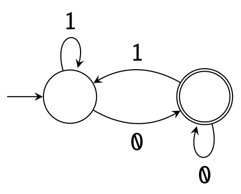
Example 1.3.1. A DFA for $\mathcal{L} = \varnothing$ is given by:



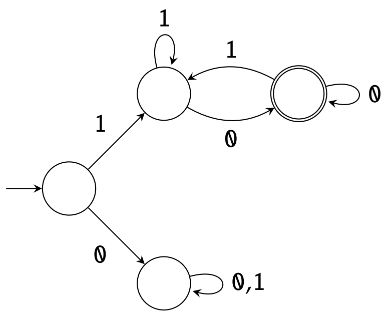
Example 1.3.2. A DFA for $\mathcal{L} = \{w : \text{every odd position is a } \mathtt{1}\}$ is given by:



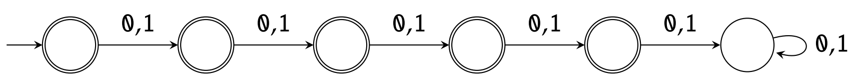
Example 1.3.3. A DFA for $\mathcal{L} = \{w : w \text{ ends in } \mathtt{0}\}$ is given by:



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



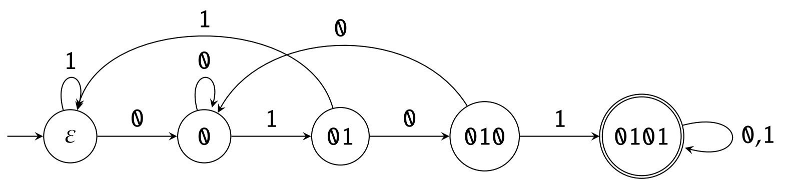
Example 1.3.5. A DFA for $\mathcal{L} = \{w : |w| \le 4\}$ is given by:



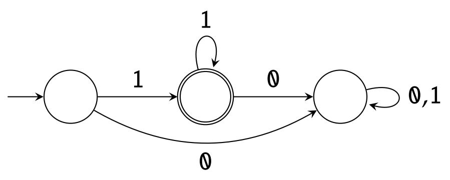
Example 1.3.6. A DFA for $\mathcal{L} = \{w : 1000 \mid |w|\}$ is given by:

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

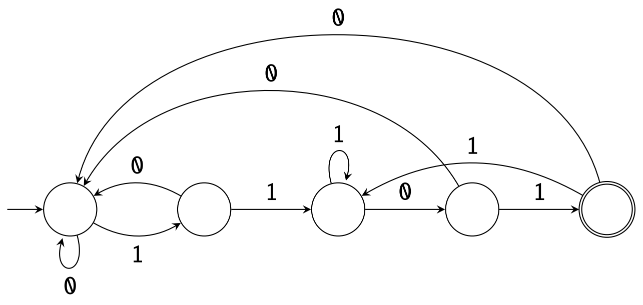
Example 1.3.7. A DFA for $\mathcal{L} = \{w : w \text{ contains } \texttt{0101} \text{ as a substring}\}$ is given by:



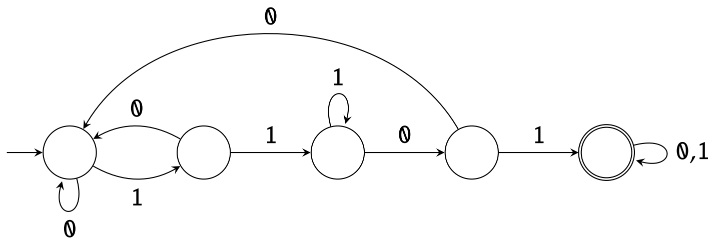
Example 1.3.8 (Week 1 Discussion). A DFA for $\mathcal{L} = \{w : |w| > 0 \land w \text{ contains only 1's}\}$ is given by:



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:



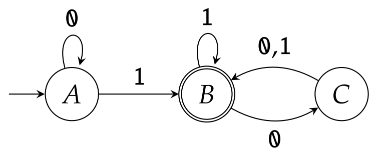
1.4 Formal definitions

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



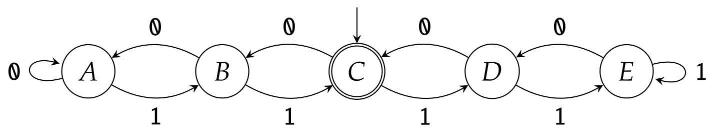
is given by $(\{A, B, C\}, \{0, 1\}, \delta, A, \{B\})$ where δ is defined by the table

	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

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

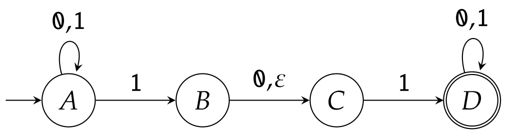
Note.

- 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: $\{\texttt{0}, \texttt{1}\}$.
- 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.

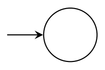
Example 2.1.3. The output table for Example 2.1.1 is given by

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

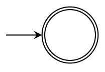
Then the language is the set of all strings containing 101 or 11.

2.2 Using shortcuts

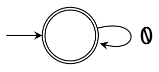
Example 2.2.1. An NFA for $\mathcal{L} = \varnothing$ is given by:



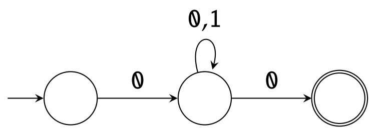
Example 2.2.2. An NFA for $\mathcal{L} = \{\varepsilon\}$ is given by:



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

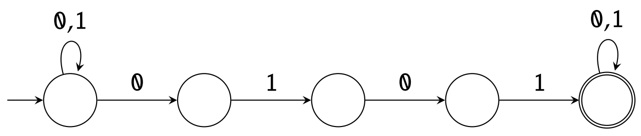


Example 2.2.4. An NFA for $\mathcal{L} = \{w : |w| \geq 2 \text{ and } w \text{ starts and ends with } 0\}$ is given by:

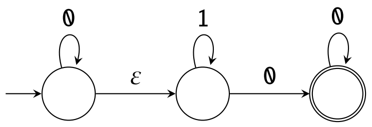


2.3 Pattern matching

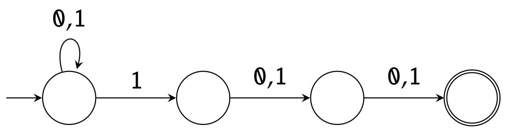
Example 2.3.1. An NFA for $\mathcal{L} = \{w : \text{conatins } \texttt{0101}\}$ is given by:



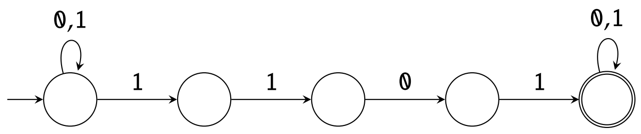
Example 2.3.2. An NFA for $\mathcal{L} = \{w : w = \underbrace{\texttt{00...0}}_{\geq 0}\underbrace{\texttt{11...1}}_{\geq 0}\underbrace{\texttt{00...0}}_{\geq 1}\}$ is given by:



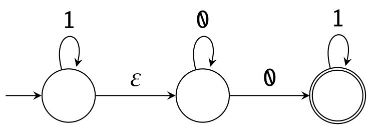
Example 2.3.3. An NFA for $\mathcal{L} = \left\{w : w \text{ has a } \texttt{1} \text{ in the 3rd position from the end}\right\}$ is given by:



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

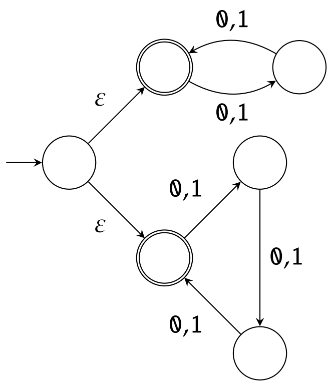


Example 2.3.5 (Week 1 Discussion). An NFA for $\mathcal{L} = \{w : w = \underbrace{\texttt{11...1}}_{\geq 0}\underbrace{\texttt{00...0}}_{\geq 1}\underbrace{\texttt{11...1}}_{\geq 0}\}$ is given by:

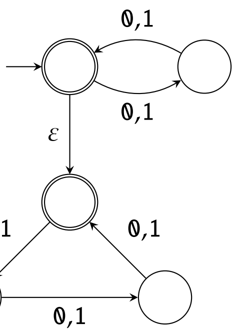


2.4 Alternatives

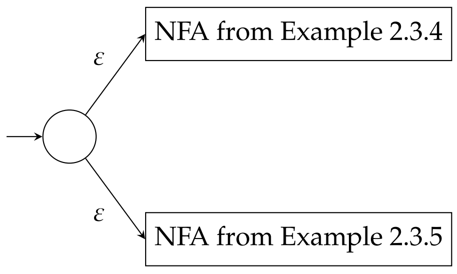
Example 2.4.1. An NFA for $\mathcal{L} = \{w : 2 \mid |w| \vee 3 \mid |w|\}$ is given by:



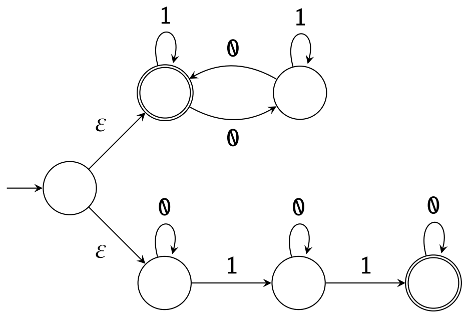
Note that the following is not valid due to side effects:



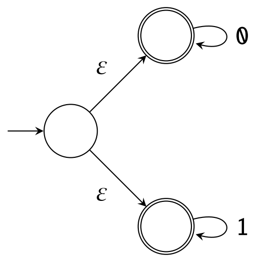
Example 2.4.2 (Week 1 Discussion). An NFA for
$\mathcal{L} = \{w : w \text{ contains } \texttt{1101} \vee w = \underbrace{\texttt{11...1}}_{\geq 0}\underbrace{\texttt{00...0}}_{\geq 1}\underbrace{\texttt{11...1}}_{\geq 0}\}$ is given by:



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



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

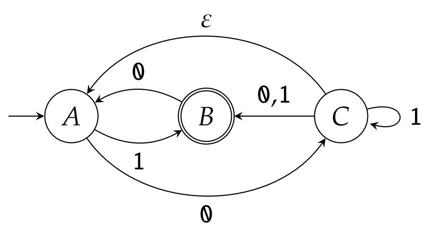
	$\mathtt{0}$	$\mathtt{1}$	ε
A	$\{A\}$	$\{A, B\}$	$\varnothing$
B	$\{C\}$	$\varnothing$	$\{C\}$
C	$\varnothing$	$\{D\}$	$\varnothing$
D	$\{D\}$	$\{D\}$	$\varnothing$.

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

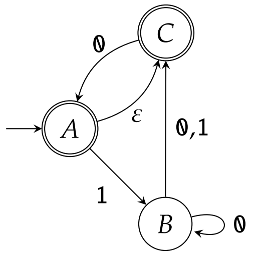
NFA N **recognizes** the language $\mathcal{L}$ iff

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

3 Equivalence of DFAs and NFAs

3.1 Example

Does this NFA accept the string `0110`?



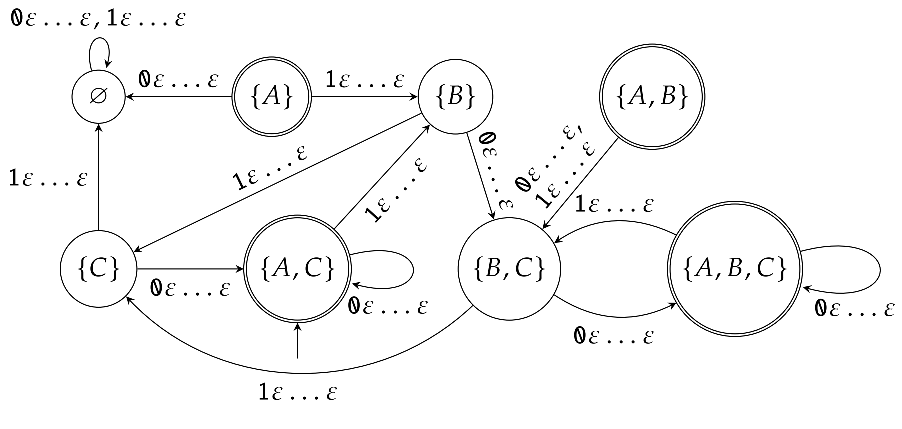
Recall that the NFA accepts `0110` iff an accept state is reachable from the start state via some path $\underbrace{\varepsilon\ldots\varepsilon}_{\geq 0}\,\mathtt{0}\,\underbrace{\varepsilon\ldots\varepsilon}_{\geq 0}\,\mathtt{1}\,\underbrace{\varepsilon\ldots\varepsilon}_{\geq 0}\,\mathtt{1}\,\underbrace{\varepsilon\ldots\varepsilon}_{\geq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

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

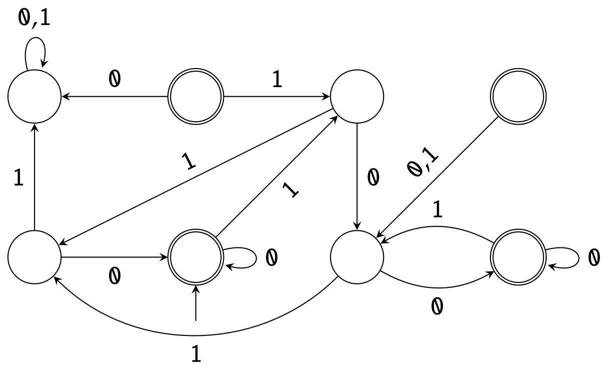
Since C is an accept state, the NFA accepts the string `0110`.

Can we convert this to a DFA? Yes.

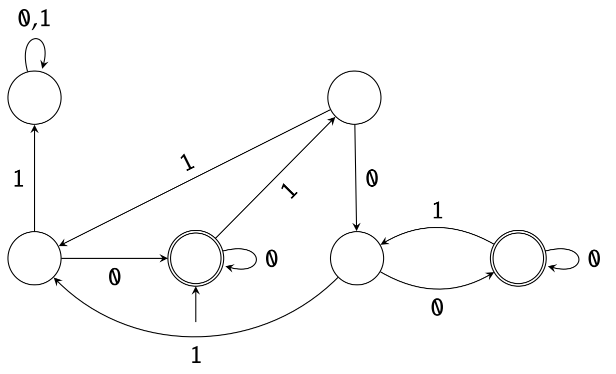
Step 1 Use subsets of the states of the NFA as the states of the DFA. Accept states are subsets that contain accept states of the NFA. The start state is the subset of all reachable states from the start state of the NFA via ε. For subsets containing more than one element, the transition is the union 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 $D = (\mathcal{P}(Q), \Sigma, \Delta, S_0, \mathscr{F})$ where

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

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

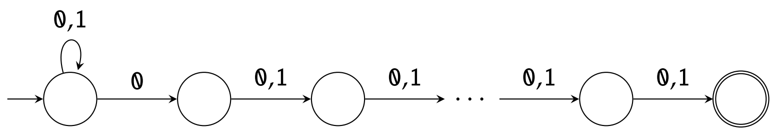
Define $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, k = any positive integer, $\mathcal{L}_k = \{w : w \text{ has a } \mathtt{0} \text{ in the } k\text{-th position from the end}\}$.

(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

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

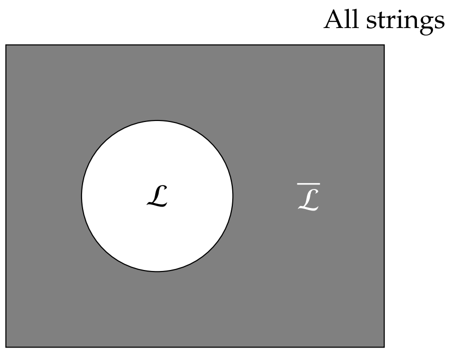
However, note that the left hand side has a `0` in the k-th position from the end while the right hand side has a `1` in the k-th position from the end.

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

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

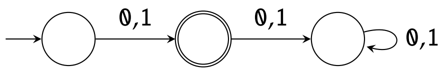
Theorem 4.1.2

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

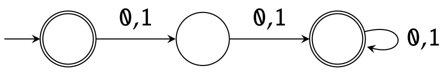
Proof. Swap the accept and reject states of the DFA for $\mathcal{L}$ to get a DFA for $\overline{\mathcal{L}}$.
Formally, if DFA $(Q, \Sigma, \delta, q_0, F)$ recognizes $\mathcal{L}$ then DFA $(Q, \Sigma, \delta, q_0, Q \setminus F)$ recognizes $\overline{\mathcal{L}}$. □

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:

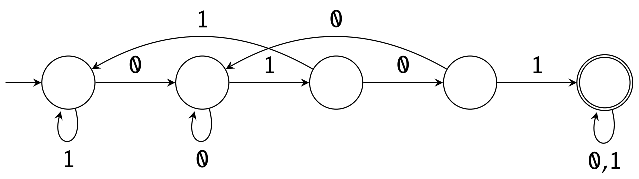


Then the DFA for $\mathcal{L}$ is given by

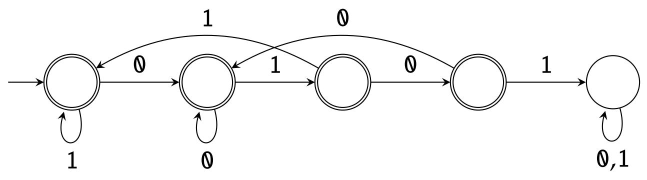


Example 4.1.4. $\mathcal{L} = \{w : w \text{ does not contain } \mathtt{0101}\}$.

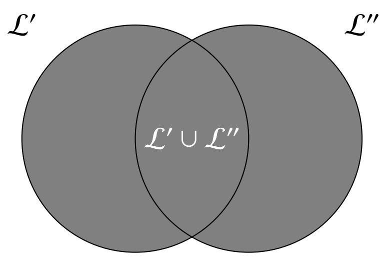
Note that $\overline{\mathcal{L}} = \{w : w \text{ contains } \mathtt{0101}\}$. The DFA for $\overline{\mathcal{L}}$ is given by:

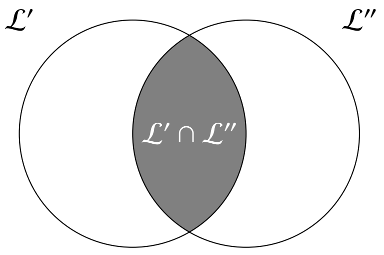


Then the DFA for $\mathcal{L}$ is given by



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.

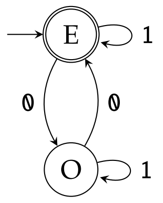
(a) Connect an initial state to the initial states for the NFAs for $\mathcal{L}'$ and $\mathcal{L}''$ via ε.

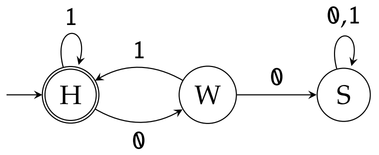
(b) If $\mathcal{L}'$ and $\mathcal{L}''$ are regular then $\overline{\mathcal{L}'}$ and $\overline{\mathcal{L}''}$ are regular. Then $\overline{\mathcal{L}'} \cup \overline{\mathcal{L}''}$ is regular. Then, by De Morgan's law, $\mathcal{L}' \cap \mathcal{L}'' = \overline{\overline{\mathcal{L}'} \cup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

□

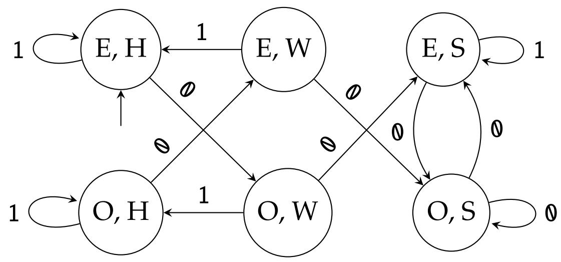
Proof. **Idea:** [1]

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



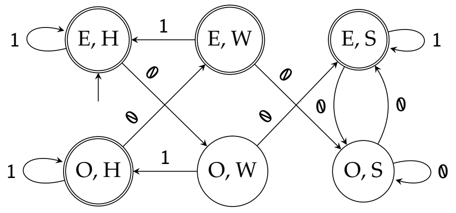


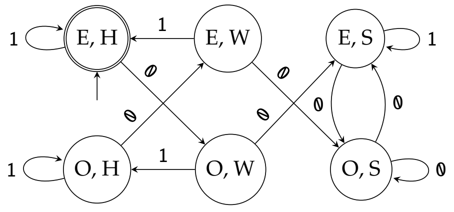
Consider the construction:



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. □

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

Corollary 4.2.5

Every finite language is regular.

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

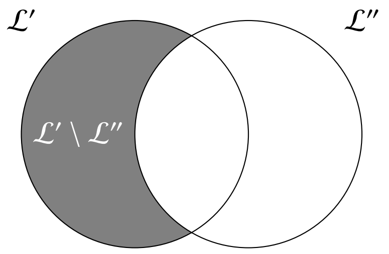
$$\mathcal{L} = \bigcup_{w \in \mathcal{L}} \{w\} ,$$

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

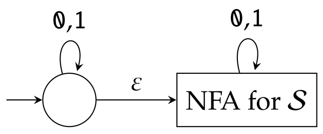
If $\mathcal{L}'$ and $\mathcal{L}''$ are regular then so is $\mathcal{L}' \setminus \mathcal{L}''$.

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 $\mathcal{L}' = \{\mathtt{000}, 111\}$ and $\mathcal{L}'' = \{1\mathtt{0}, \mathtt{0}1\mathtt{0}\}$, we have

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

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:

$$\begin{tikzcd} {} \arrow[r] & \boxed{\text{NFA for } \mathcal{L}'} \arrow[r, "\varepsilon"] & \boxed{\text{NFA for } \mathcal{L}''} \end{tikzcd}$$

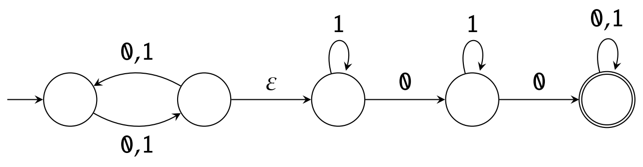
where the ε transition from the first NFA to the second NFA refer to those from the accept states in the first NFA to the start state of the second and all accept states in the first NFA are converted to reject states.

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''.

$\square$

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 $\mathcal{L}^*$.

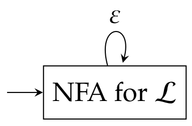
Proof. (Incorrect) Note that

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

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. □

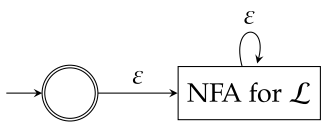
Proof. (Incorrect) An NFA for $\mathcal{L}^*$ is given by



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 $w = w_1 w_2 \ldots w_n$, its **reverse** is given by

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

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

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

Definition 4.7.2

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

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

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

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

4.9 Shuffle

Definition 4.9.1

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

$$\left\{a_1 b_1 a_2 b_2 \ldots a_k b_k : a_1, \ldots, a_k \text{ strings}, b_1, \ldots, b_k \text{ strings}, a_1 \cdots a_k \in \mathcal{L}', b_1 \cdots b_k \in \mathcal{L}''\right\}.$$

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

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

□

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 $\Sigma^* \cup \{\mathtt{1}\} = \Sigma^*$.

Example 5.1.6. Example 5.1.3 can be simplified to $\mathtt{0111}$.

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

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

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

Example 5.1.10. A regular expression for $\{w : w \text{ contains } \mathtt{001} \text{ as a substring}\}$ is $\Sigma^*\mathtt{001}\Sigma^*$.

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

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 $\Sigma \cup \mathtt{0}\Sigma^*\mathtt{0} \cup \mathtt{1}\Sigma^*\mathtt{1}$.

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 $(\mathtt{0} \cup \varepsilon)\mathtt{1}^*$ can be arguably simplified to $\mathtt{0}\mathtt{1}^* \cup \mathtt{1}^*$.

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 $(0 \cup 1^+0)^*$ can be simplified to $((\varepsilon \cup 1^+)0)*$ and then to $(1^*0)^*$. In words, all strings that do not end in a 1.

Example 5.1.20.

Regular expression	Shortest accepted string	Shortest rejected string
0^*0011^*	001	ε
$0^*00(0 \cup 1)^*$	00	ε
$(00)^*(110)^*(11)^*$	ε	0, 1
$0^*(01 \cup 10)^*1^*$	ε	100, 110

5.2 Equivalence with automata

Theorem 5.2.1

The language of every regular expression is regular.

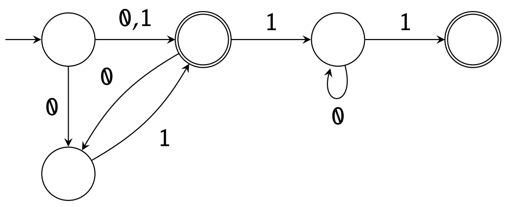
Proof. Induction. □

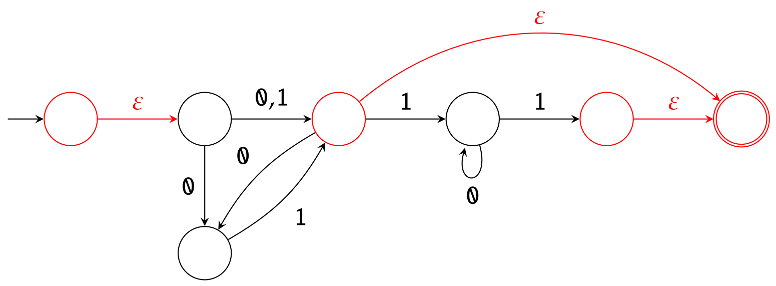
Theorem 5.2.2: Kleene's theorem

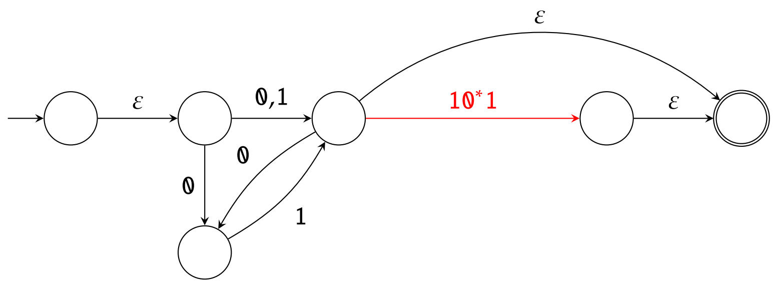
Then language of every NFA can be represented by a regular expression.

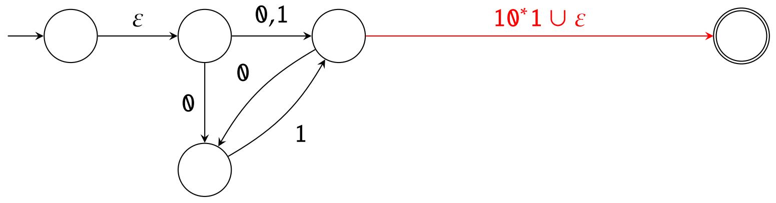
Proof. Given an NFA, create a new state state and a new and unique accept state (making every state in the original NFA rejecting). Choose a state other than the start state and the accept state. Call the states from which the incoming transitions originate donors and those at which the outgoing transitions 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. □

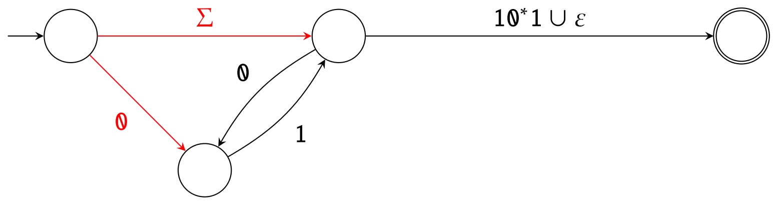
5.3 Conversion example

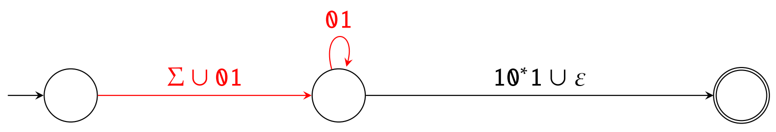


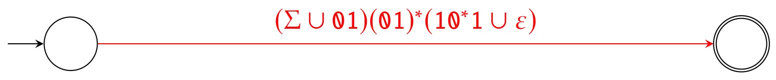




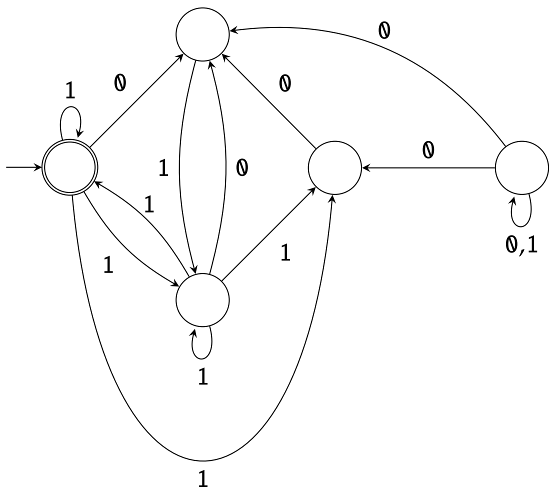


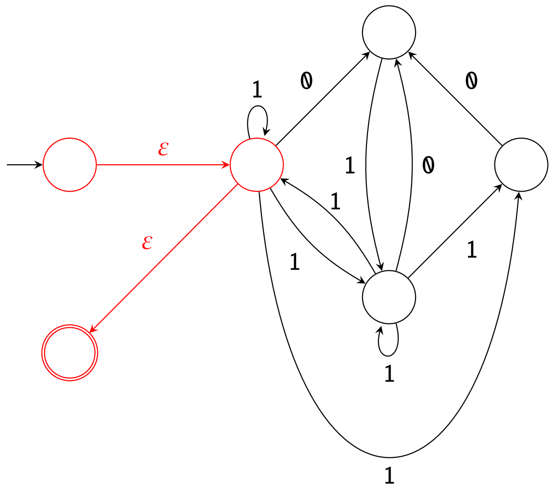


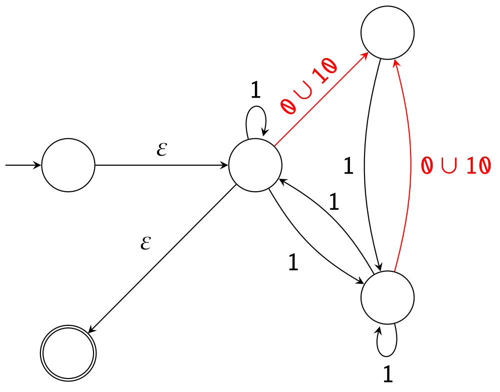


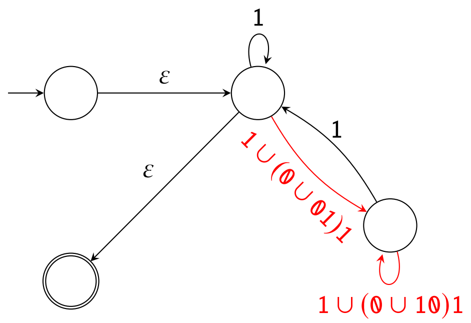

$(\Sigma \cup 01)(01)^*(10^*1 \cup \varepsilon)$

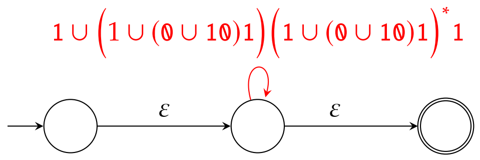
5.4 Conversion example #2

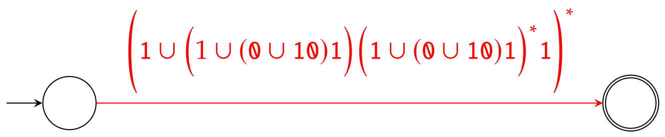







1
ε
1
ε
1 ∪ (∅ ∪ ∅1)1
1 ∪ (∅ ∪ 1∅)1


$1 \cup \left(1 \cup (\emptyset \cup 1\emptyset)1\right)\left(1 \cup (\emptyset \cup 1\emptyset)1\right)^* 1$
ε
ε


$\left(1 \cup \left(1 \cup (\emptyset \cup 1\emptyset)1\right)\left(1 \cup (\emptyset \cup 1\emptyset)1\right)^* 1\right)^*$

6 Pumping lemma

6.1 First nonregular language

Theorem 6.1.1

The language $\mathcal{L} = \{\mathtt{0}^n \mathtt{1}^n : n \geq 1\}$ is not regular.

Proof. By contradiction. Suppose we have a DFA D for $\mathcal{L}$. Let p be the number of states in D.

- Run the DFA on $w = \underbrace{\mathtt{00...0}}_{p}\underbrace{\mathtt{11...1}}_{p}$.
- Note that w must end in an accept state.
- By the pigeonhole principle, some state q must have been visited at least twice while reading $\underbrace{\mathtt{00...0}}_{p}$.
- Write $w = xyz$ where x is the portion of w before the first visit of q, y is the portion of w between the first and the second visit of q, and z is the portion of w after the second visit of q.
- Since D accepts $w = xyz$, D also accepts $xyyz$, which contains more than p $\mathtt{0}$'s and exactly p $\mathtt{1}$'s. Then D does not recognize $\mathcal{L}$.

□

6.2 Pumping lemma

Lemma 6.2.1

Let $\mathcal{L}$ be a regular language. Then there exists $p \in \mathbb{N}$ such that every string $w \in \mathcal{L}$ of length $\geq p$ can be written as

$$w = xyz,$$

where

- x, y, z are strings,
- $y \neq \varepsilon$,
- $|xy| \leq p$, and
- $xy^iz \in \mathcal{L}$ for $i = 0, 1, 2, \ldots$.

To put in a slightly irresponsible wording, all long enough strings in $\mathcal{L}$ can be pumped.

Proof. Let D be a DFA for $\mathcal{L}$. Define p to be the number of states in D.

- Run the DFA on w.
- Note that w must end in an accept state.
- By the pigeonhole principle, some state q must have been visited at twice while reading $w_1w_2\ldots w_p$.
- Define x to be the portion of w before the first visit of q, y to be the portion of w between the first and the second visit of q, and z to be the portion of w after the second visit of q.
- Note that x, y, and z are strings, $y \neq \varepsilon$, $|xy| \leq |w_1w_2\ldots w_p| = p$, and $xy^iz \in \mathcal{L}$ for all i.

□

Original pumping lemma ("pumping property")

$$\mathcal{L} \text{ reg} \Rightarrow (\exists p \, \forall w \in \mathcal{L} : |w| \geq p \Rightarrow \exists x, y, z : w = xyz \wedge y \neq \varepsilon \wedge |xy| \leq p \wedge \forall i : xy^iz \in \mathcal{L}).$$

Contrapositive form of pumping lemma

$$(\forall p \, \exists w \in \mathcal{L} : |w| \geq p \Rightarrow \forall x, y, z : w = xyz \wedge y \neq \varepsilon \wedge |xy| \leq p \Rightarrow \exists i : xy^iz \notin \mathcal{L}) \Rightarrow \mathcal{L} \text{ not regular}$$

6.3 Examples

Example 6.3.1. Consider the language $\mathcal{L} = \{\mathtt{0}^n 1^n : n \geq 0\}$.

Let p be arbitrary. Define $w = \mathtt{0}^p 1^p \in \mathcal{L}$. Consider arbitrary x, y, z such that $w = xyz$, $y \neq \varepsilon$, and $|xy| \leq p$. Note that $xy^2z \notin \mathcal{L}$.

Then the pumping property fails and $\mathcal{L}$ is not regular.

Example 6.3.2. Consider the language $\mathcal{L} = \{w : w \text{ contains as many } \mathtt{0}\text{'s as } 1\text{'s}\}$.

Note that

$$\mathcal{L} \cap \underbrace{\mathtt{0}^* 1^*}_{\text{regular}} = \underbrace{\{\mathtt{0}^n 1^n : n \geq 0\}}_{\text{nonregular}}.$$

Then by the closure of regular languages under intersection, $\mathcal{L}$ is not regular.

Example 6.3.3. Consider the language $\mathcal{L} = \{\mathtt{0}^n 1^m : n \neq m\}$.

Note that

$$\underbrace{\mathtt{0}^* 1^*}_{\text{regular}} \setminus \mathcal{L} = \underbrace{\{\mathtt{0}^n 1^n : n \geq 0\}}_{\text{nonregular}}.$$

Then by the closure of regular languages under set difference, $\mathcal{L}$ is not regular.

Example 6.3.4. Consider the language $\mathcal{L} = \{\mathtt{0}^n 1^m : n > m\}$ and the string $\overbrace{\mathtt{00}\ldots\mathtt{0}}^{p+1}\overbrace{11\ldots1}^{p}$. Consider any $y \neq \varepsilon$ such that $|xy| \leq p$. Note that $xz \notin \mathcal{L}$. Then $\mathcal{L}$ is not regular.

Example 6.3.5. Consider the language $\mathcal{L} = \{1\mathtt{0}^n 1^n : n \geq 1\}$ and the string $1\overbrace{\mathtt{00}\ldots\mathtt{0}}^{p}\overbrace{11\ldots1}^{p}$. Then $y \neq \varepsilon$ either contains a leading 1, in which case xz does not have a leading 1 and therefore $xz \notin \mathcal{L}$, or does not contain a leading 1, in which case xz has a different number of $\mathtt{0}$'s as trailing 1's and therefore $xz \notin \mathcal{L}$. Then $\mathcal{L}$ is not regular.

Example 6.3.6. Consider the language $\mathcal{L} = \left\{1^{n^2} : n \geq 0\right\}$ and the string $\overbrace{11\ldots1}^{p^2}$. Consider any $y \neq \varepsilon$ such that $|xy| \leq p$. Then

$$\begin{aligned} p^2 < |xy^2z| &= |xyz| + |y| \\ &\leq p^2 + p < (p+1)^2. \end{aligned}$$

Then $xy^2z \notin \mathcal{L}$ and $\mathcal{L}$ is not regular.

Example 6.3.7. Consider the language $\mathcal{L} = \{ww : w \in \{\mathtt{0}, 1\}^*\}$ and the string $\overbrace{\mathtt{00}\ldots\mathtt{0}}^{p}1\overbrace{\mathtt{00}\ldots\mathtt{0}}^{p}1$. Consider any $y \neq \varepsilon$ such that $|xy| \leq p$. Note that $xz \notin \mathcal{L}$ since the first half of xz ends with $\mathtt{0}$ while the second half ends with 1. Then $\mathcal{L}$ is not regular.

Example 6.3.8. Consider the language $\mathcal{L} = \{www : w \in \{\mathtt{0}, 1\}^*\}$. It can be shown that this is not regular.

Example 6.3.9. Consider the language $\mathcal{L} = \left\{w : w = w^R\right\}$ and the string $\overbrace{\texttt{00...0}}^{p}\texttt{1}\overbrace{\texttt{00...0}}^{p}$. Consider any $y \neq \varepsilon$ such that $\left|xy\right| \leq p$. Note that $xz \notin \mathcal{L}$. Then $\mathcal{L}$ is not regular.

Example 6.3.10. Consider the language $\mathcal{L} = \left\{w : |w| \text{ prime}\right\}$. Let q be a prime greater than p. Consider the string $\overbrace{\texttt{00...0}}^{q}$. Consider any $y \neq \varepsilon$ such that $\left|xy\right| \leq p$. Then $xy^{q+1}z \notin \mathcal{L}$ since $\left|xy^{q+1}z\right| = \underbrace{q}_{\geq 2}\underbrace{(1 + \left|y\right|)}_{\geq 2}$ is not prime. Then $\mathcal{L}$ is not regular.

6.4 True or false

Example 6.4.1. The statement "regular languages are closed under arbitrary unions, i.e., even infinite unions" is false. Note that every nonregular language is an infinite union of singleton languages, which are regular. For a concrete example, note that $\{\mathtt{0}^n\mathtt{1}^n : n \geq 0\} = \bigcup_{n=0}^{\infty}\{\mathtt{0}^n\mathtt{1}^n\}$.

Example 6.4.2. The statement "subsets of regular languages are regular" is false. Note that $\Sigma*$ is regular yet $\{\mathtt{0}^n\mathtt{1}^n : n \geq 0\} \subseteq \Sigma^*$.

Example 6.4.3. The statement "subsets of nonregular languages are nonregular" is false. Note that $\varnothing$ is regular yet $\varnothing \subseteq \{\mathtt{0}^n\mathtt{1}^n : n \geq 0\}$.

Example 6.4.4. The statement "nonregular languages are closed under complement" is true since $\mathcal{L}$ regular $\Leftrightarrow$ $\overline{\mathcal{L}}$ regular, which implies $\mathcal{L}$ nonregular $\Leftrightarrow$ $\overline{\mathcal{L}}$ nonregular.

Example 6.4.5. The statement "nonregular languages are closed under union" is false. Note that $\mathcal{L} = \{\mathtt{0}^n\mathtt{1}^n : n \geq 0\}$ is nonregular and so is $\overline{\mathcal{L}}$, yet $\mathcal{L} \cup \overline{\mathcal{L}} = \Sigma^*$ is regular.

Example 6.4.6. The statement "nonregular languages are closed under intersection" is false. Note that $\mathcal{L} = \{\mathtt{0}^n\mathtt{1}^n : n \geq 0\}$ is nonregular and so is $\overline{\mathcal{L}}$, yet $\mathcal{L} \cap \overline{\mathcal{L}} = \varnothing$ is regular.

Example 6.4.7. The statement "nonregular languages are always infinite" is true since all finite languages are regular.

7 Myhill-Nerode theorem

7.1 When the pumping lemma fails

Example 7.1.1. Consider the language $\mathcal{L} = \mathtt{0}^*\mathtt{1}^* \cup \{\mathtt{1}^m\mathtt{0}^n\mathtt{1}^n : m \geq 1 \land n \geq 0\}$. We can consider strings of the forms $\mathtt{0}^+$, $\mathtt{1}^+$, $\mathtt{0}^+\mathtt{1}^+$, $\mathtt{1}^+\mathtt{0}^n\mathtt{1}^n$, etc. We can check that all of the listed forms satisfy the pumping property by taking $x = \varepsilon$ and $y = w_1$. Then in this case the pumping lemma is useless. However $\mathcal{L}$ is nonregular since $\mathcal{L} \cap \mathtt{1}\mathtt{0}^+\mathtt{1}^+ = \{\mathtt{1}\mathtt{0}^n\mathtt{1}^n : n \geq 1\}$ is nonregular.

7.2 Indistinguishability

Definition 7.2.1

Let $\mathcal{L}$ be a given language. Strings x and y are called **$\mathcal{L}$-indistinguishable** if for all strings w,

$$xw \in \mathcal{L} \Leftrightarrow yw \in \mathcal{L}.$$

Notation: $x \equiv_{\mathcal{L}} y$

Example 7.2.2. Consider the language $\mathcal{L} = \{w : 3 \mid |w|\}$. Note that

- $1 \equiv_{\mathcal{L}} 0$,
- $\varepsilon \not\equiv_{\mathcal{L}} 0$,
- $\varepsilon \equiv_{\mathcal{L}} 000$, and
- $1 \equiv_{\mathcal{L}} 1010 \equiv_{\mathcal{L}} 1110001$.

Example 7.2.3. Consider the language $\mathcal{L} = \{w : 3 \mid \sum w_i\}$. Note that

- $1 \not\equiv_{\mathcal{L}} 0$,
- $\varepsilon \equiv_{\mathcal{L}} 0 \equiv_{\mathcal{L}} 000$,
- $1 \equiv_{\mathcal{L}} 101011$

Proposition 7.2.4

For any language $\mathcal{L}$, $\equiv_{\mathcal{L}}$ is an equivalence relation.

Proof.

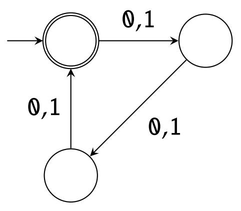
- Reflexive: $x \equiv_{\mathcal{L}} x$.
- Symmetric: $x \equiv_{\mathcal{L}} y \Leftrightarrow y \equiv_{\mathcal{L}} x$.
- Transitive: $xw \in \mathcal{L} \Leftrightarrow yw \in \mathcal{L} \Leftrightarrow zw \in \mathcal{L}$.

□

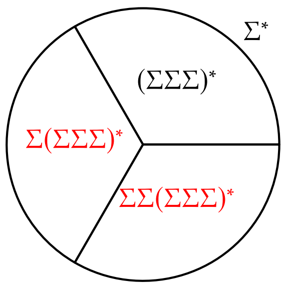
Note. Every equivalence class is either entirely in $\mathcal{L}$ or entirely in $\overline{\mathcal{L}}$.

Example 7.2.5. Consider the language $\mathcal{L} = \{w : 3 \mid |w|\}$.

A DFA for $\mathcal{L}$ is given by:

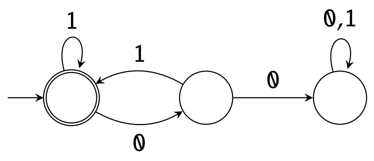


Then the language can be partitioned into:

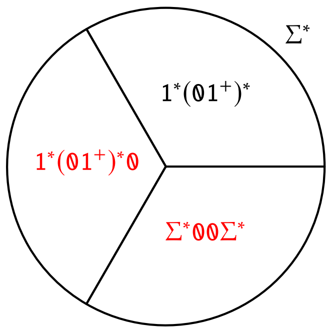

Σ^*
$(\Sigma\Sigma\Sigma)^*$
$\Sigma(\Sigma\Sigma\Sigma)^*$
$\Sigma\Sigma(\Sigma\Sigma\Sigma)^*$

Example 7.2.6. Consider the language $\mathcal{L} = \{w : \text{every } \mathtt{0} \text{ is immediately followed by a } \mathtt{1}\}$.

A DFA for $\mathcal{L}$ is given by:

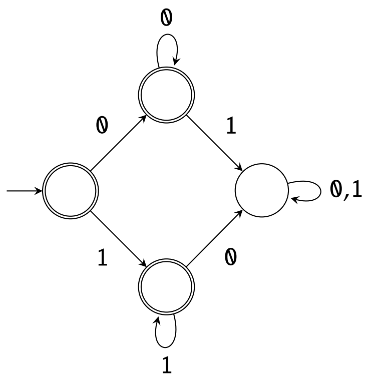


Then the language can be partitioned into:

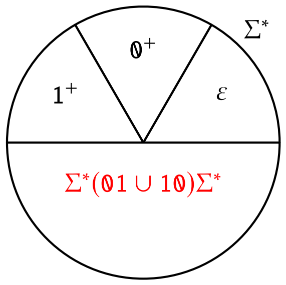

Σ^*
$1^*(\mathtt{0}1^+)^*$
$1^*(\mathtt{0}1^+)^*\mathtt{0}$
$\Sigma^*\mathtt{00}\Sigma^*$

Example 7.2.7. Consider the language $\mathcal{L} = \{w : w \text{ does not contain both a } \mathtt{0} \text{ and a } \mathtt{1}\}$.

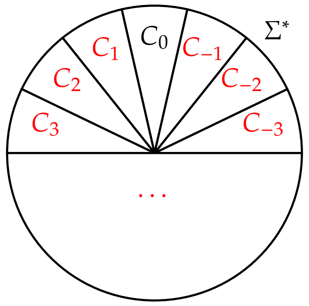
A DFA for $\mathcal{L}$ is given by:



Then the language can be partitioned into:



Example 7.2.8. Consider the language $\mathcal{L} = \{w : w \text{ contains as many } \mathtt{0}\text{'s as } 1\text{'s}\}$. Note that this language is nonregular. The language can be partitioned into:



where

$$C_i = \{w : \text{number of } \mathtt{0}\text{'s in } w - \text{number of } 1\text{'s in } w = i\}.$$

Example 7.2.9. Consider the language $\mathcal{L} = \{w : w^R = w\}$. Then

$$w \neq w' \Rightarrow w\mathtt{0}^n 1\mathtt{0}^n w^R \in \mathcal{L} \wedge w'\mathtt{0}^n 1\mathtt{0}^n w^R \notin \mathcal{L}$$

for large enough n. Then every $w \in \Sigma^*$ is in its own equivalence class.

7.3 Myhill-Nerode theorem

Theorem 7.3.1

For every language $\mathcal{L}$, the minimum number of states in a DFA for $\mathcal{L}$ is equal to the number of equivalence classes under $\equiv_{\mathcal{L}}$.

If this number is infinite, we interpret it as "no such DFA exists," that is, $\mathcal{L}$ is nonregular.

Proof. $\geq$: By contradiction.

Fix a DFA for $\mathcal{L}$ with k states. Suppose $\exists\, x_1, x_2, \ldots, x_{k+1}$, each in a different equivalence class of $\equiv_{\mathcal{L}}$.

Input string	End state
x_1	q_{27}
x_2	q_6
x_3	q_5
$\vdots$	$\vdots$
x_{k+1}	q_{12}

Note that the left hand side of the table enumerates $k+1$ strings while the right hand side only has k possibilities. Then for some x_i, x_j where $i \neq j$, the DFA ends in the same state on reading x_i as x_j. Then for all strings w we have

$$x_i w \in \mathcal{L} \Leftrightarrow x_j w \in \mathcal{L},$$

that is,

$$x_i \equiv_{\mathcal{L}} x_j,$$

a contradiction.

$\leq$: Suppose $\mathcal{L}$ has k equivalence classes. Pick a representative from each class: $S_1 = \varepsilon, S_2, \ldots, S_k$. Then a DFA for $\mathcal{L}$ is given by

$$(\{S_1, S_2, \ldots, S_k\}, \Sigma, S_1, \delta, \{S_1, S_2, \ldots, S_k\} \cap \mathcal{L})$$

where

$$\delta(S_i, \sigma) = \text{representative for the equivalence class of } S_i\sigma.$$

□

7.4 Proving nonregularity

There are 3 methods to prove nonregularity:

1. Pumping lemma,
2. Pumping lemma with closure properties, and
3. Myhill-Nerode theorem.

Example 7.4.1. Consider the languages $\mathcal{L}_1 = \left\{w : w^R\right\}$ and $\mathcal{L}_2 = \left\{w : w \text{ has equally many } \mathtt{0}\text{'s and } \mathtt{1}\text{'s}\right\}$. we have shown that $\equiv_{\mathcal{L}_1}$ and $\equiv_{\mathcal{L}_2}$ have infinitely many equivalence classes. Then $\mathcal{L}_1$ and $\mathcal{L}_2$ are nonregular.

Example 7.4.2. Consider the language $\mathcal{L} = \{ww : w \in \{\mathtt{0}, \mathtt{1}\}^*\}$. Note that

$$i \neq j \Longrightarrow \mathtt{0}^i\mathtt{1}\mathtt{0}^i\mathtt{1} \in \mathcal{L} \wedge \mathtt{0}^j\mathtt{1}\mathtt{0}^i\mathtt{1} \notin \mathcal{L}.$$

Then $\varepsilon, \mathtt{0}, \mathtt{00}, \mathtt{000}, \ldots$ are in different classes. Then $\equiv_{\mathcal{L}}$ has infinitely many classes. Then $\mathcal{L}$ is nonregular.

Example 7.4.3. Consider the language $\mathcal{L} = \{\mathtt{0}^n\mathtt{1}^m : n \neq m\}$. Note that

$$i \neq j \Longrightarrow \mathtt{0}^i\mathtt{1}^j \in \mathcal{L} \wedge \mathtt{0}^j\mathtt{1}^j \notin \mathcal{L} \Longrightarrow \mathtt{0}^i \not\equiv_{\mathcal{L}} \mathtt{0}^j.$$

Then $\varepsilon, \mathtt{0}, \mathtt{00}, \mathtt{000}, \ldots$ are in different classes. Then $\equiv_{\mathcal{L}}$ has infinitely many classes. Then $\mathcal{L}$ is nonregular.

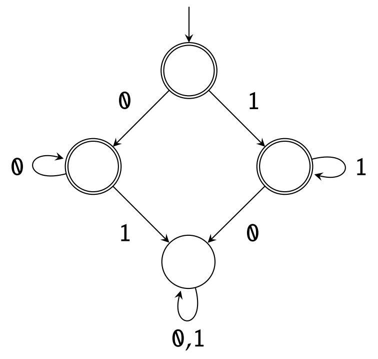
Example 7.4.4. Consider the language $\mathcal{L} = \mathtt{0}^*\mathtt{1}^* \cup \bigcup_{n=0}^{\infty} \mathtt{1} * \mathtt{0}^n\mathtt{1}^n$. The nonregularity of this language was not provable by using the pumping lemma. However, note that

$$i \neq j \Longrightarrow \mathtt{1}\mathtt{0}^i\mathtt{1}^i \in \mathcal{L} \wedge \mathtt{1}\mathtt{0}^j\mathtt{1}^i \notin \mathcal{L} \Longrightarrow \mathtt{1}\mathtt{0}^i \not\equiv_{\mathcal{L}} \mathtt{1}\mathtt{0}^j.$$

Then $\mathtt{10}, \mathtt{100}, \mathtt{1000}, \ldots$ are in different classes. Then $\equiv_{\mathcal{L}}$ has infinitely many classes. Then $\mathcal{L}$ is nonregular.

7.5 Proving optimality

Example 7.5.1. We claim that the smallest possible DFA for the language $\mathtt{0}^* \cup \mathtt{1}^*$ is given by:

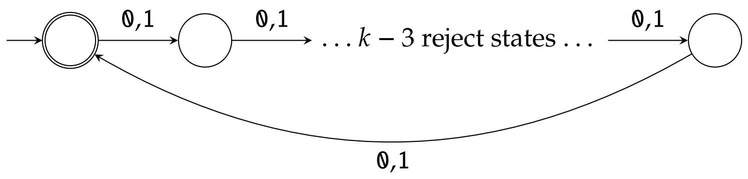


Consider the following table of distinguishing suffixes:

	ε	0	1	01
ε				
0	1			
1	0	0		
01	ε	ε	ε	

Then $\varepsilon, \mathtt{0}, \mathtt{1}, \mathtt{01}$ are in different classes, that is, there are at least 4 equivalence classes. Then the minimum size of any DFA for $\mathcal{L}$ is 4. Then our DFA is optimal.

Example 7.5.2. Find the smallest possible DFA for $\mathcal{L}_k = \{w : k \mid |w|\}$.



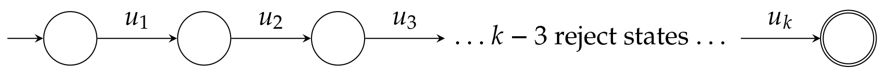
To show that no smaller DFA exists, note that

$$i \neq j (\mathrm{mod} k) \Longrightarrow \mathtt{0}^i \mathtt{0}^{k-i} \in \mathcal{L}_k \wedge \mathtt{0}^j \mathtt{0}^{k-i} \notin \mathcal{L}_k.$$

Corollary 7.5.3

For each k, $\mathcal{L}_k$ is a language that can be recognized by a DFA with k states but not $k-1$ states.

Example 7.5.4. Given $u = u_1 u_2 \cdots u_k$, define $\mathcal{L} = \Sigma^* u_1 u_2 \cdots u_k \Sigma^*$. Find the smallest possible DFA for $\mathcal{L}$.



Note that one still needs to figure out the transitions for $\neg u_1, \neg u_2, \ldots \neg u_k$ from the respective nodes. However, to show that there exists no smaller DFA for $\mathcal{L}$, let $i, h \in \{0, 1, \ldots, k\}$ where $i > j$. Note that

$$u_1 u_2 \cdots u_i u_{i+1} \cdots u_k \in \mathcal{L} \wedge \underbrace{u_1 u_2 \cdots u_j u_{i+1} \cdots u_k}_{<k} \notin \mathcal{L}.$$

Then $\varepsilon, u_1, u_1 u_2, \ldots, u_1 u_2 \cdots u_k$ are in different classes of $\equiv_{\mathcal{L}}$. Then $\mathcal{L}$ has at least $k + 1$ classes.

8 Context-free grammars

8.1 Context-free grammars defined

Example 8.1.1. A context-free grammar has an alphabet, for brevity we use $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ here. These are called "terminals" (alphabet symbols).

The second component of a grammar is a set of variables $V = \{S, A, B\}$. These are called "nonterminals."

Then there are what are known as substitution rules, for instance,

$$\begin{aligned} S &\longrightarrow A \\ S &\longrightarrow B \\ A &\longrightarrow \mathtt{a}A\mathtt{b} \\ A &\longrightarrow \varepsilon \\ B &\longrightarrow \mathtt{b}B\mathtt{a} \\ B &\longrightarrow \varepsilon. \end{aligned}$$

Here S is denoted the start variable.

How is this a computational device? What strings does this accept? What strings does this reject?

It all comes down to *derivation*:

$$\begin{aligned} S &\longrightarrow A && \text{rule \#1} \\ &\longrightarrow \mathtt{a}A\mathtt{b} && \text{rule \#3} \\ &\longrightarrow \mathtt{aa}A\mathtt{bb} && \text{rule \#3} \\ &\longrightarrow \mathtt{aaa}A\mathtt{bbb} && \text{rule \#3} \\ &\longrightarrow \mathtt{aaabbb} && \text{rule \#4} \end{aligned}$$

It a string can be derived by some application of the rule, then the string is in the language, and vice versa.

Definition 8.1.2

The **language** of a context-free grammar (CFG) is the set of all strings that can be derived.

Shorthand:

$$\begin{aligned} S &\longrightarrow A \mid B \\ A &\longrightarrow \mathtt{a}A\mathtt{b} \mid \varepsilon \\ B &\longrightarrow \mathtt{b}B\mathtt{a} \mid \varepsilon \end{aligned}$$

The language is $\mathcal{L} = \{\mathtt{a}^n\mathtt{b}^n \cup \mathtt{b}^n\mathtt{a}^n : n \geq 0\}$.

Definition 8.1.3

A **context-free grammar** is a tuple (V, Σ, R, S) where

- V = a set of variables,
- Σ = the alphabet,
- R = the set of rules, and
- S = the start variable.

Then the CFG we discussed above can be written as

$$(\{S, A, B\}, \{\mathtt{a}, \mathtt{b}\}, \{S \to A \mid B, A \to \mathtt{a}A\mathtt{b} \mid \varepsilon, B \to \mathtt{b}B\mathtt{a} \mid \varepsilon\}, S).$$

8.2 Reading a CFG

Example 8.2.1. Consider the CFG with rules

$$\begin{aligned} S &\to AA \\ A &\to A. \end{aligned}$$

The language is $\varnothing$.

Example 8.2.2. Consider the CFG with a single rule

$$S \to \mathtt{ab}S \mid \mathtt{a}.$$

The language is $(\mathtt{ab})^*\mathtt{a}$, or equivalently $\mathtt{a}(\mathtt{ba})^*$.

Example 8.2.3. Consider the CFG with a single rule

$$S \to \mathtt{a}S\mathtt{b} \mid \varepsilon.$$

The language is $\{\mathtt{a}^n\mathtt{b}^n : n \geq 0\}$.

Example 8.2.4. Consider the CFG with rules

$$\begin{aligned} S &\to \mathtt{a}S\mathtt{a} \mid B \\ B &\to \mathtt{b}B \mid \varepsilon. \end{aligned}$$

The language is $\{\mathtt{a}^n\mathtt{b}^m\mathtt{a}^n : n \geq 0 \wedge m \geq 0\}$ or $\mathtt{a}^n\mathtt{b}^*\mathtt{a}^n$ where $n \geq 0$.

Example 8.2.5. Consider the CFG with a single rule

$$S \to SS \mid \mathtt{a}S \mid \mathtt{b}.$$

The language is $\Sigma^*\mathtt{b}$.

Example 8.2.6. Consider the CFG with rules

$$\begin{aligned} S &\to AB \\ A &\to \mathtt{a}A\mathtt{a} \mid \varepsilon \\ B &\to \mathtt{b}B\mathtt{b} \mid \varepsilon. \end{aligned}$$

The language is $(\mathtt{aa})^*(\mathtt{bb})^*$.

Example 8.2.7. Consider the CFG with rules

$$\begin{aligned} S &\to \varepsilon \mid \mathtt{a}T \mid \mathtt{b}T \\ T &\to \mathtt{a}S \mid \mathtt{b}S. \end{aligned}$$

The language is $(\Sigma\Sigma)^*$.

8.3 Constructing CFGs

Example 8.3.1. Consider the language $\mathcal{L} = \{\mathtt{a}^n(\mathtt{a} \cup \mathtt{b})^n : n \geq 0\}$.
A CFG for this language is given by

$$S \longrightarrow \mathtt{a}S\mathtt{a} \mid \mathtt{a}S\mathtt{b} \mid \varepsilon.$$

Example 8.3.2. Consider the language $\mathcal{L} = \{(\mathtt{a} \cup \mathtt{b})^n\mathtt{a}(\mathtt{a} \cup \mathtt{b})^n : n \geq 0\}$.
A CFG for this language is given by

$$\begin{aligned} S &\longrightarrow TST \mid \mathtt{a} \\ T &\longrightarrow \mathtt{a} \mid \mathtt{b}. \end{aligned}$$

Example 8.3.3. Consider the language $\mathcal{L} = \{\mathtt{a}^n\mathtt{b}^m : 0 \leq m \leq 3n\}$.
A CFG for this language is given by

$$\begin{aligned} S &\longrightarrow \mathtt{a}SBBB \mid \varepsilon \\ B &\longrightarrow \mathtt{b} \mid \varepsilon. \end{aligned}$$

Example 8.3.4. Consider the language $\mathcal{L} = \left\{w : w = w^R\right\}$.
A CFG for this language is given by

$$S \longrightarrow \mathtt{a}S\mathtt{a} \mid \mathtt{b}S\mathtt{b} \mid \mathtt{a} \mid \mathtt{b} \mid \varepsilon.$$

Example 8.3.5. Consider the language $\mathcal{L} = \{\mathtt{a}^n\mathtt{b}^m : n \geq m\}$.
A CFG for this language is given by

$$S \longrightarrow \mathtt{a}S \mid \mathtt{a}S\mathtt{b} \mid \varepsilon.$$

Example 8.3.6. Consider the language $\mathcal{L} = \{\mathtt{a}^n\mathtt{b}^m : n > m\}$.
A CFG for this language is given by

$$S \longrightarrow \mathtt{a}S \mid \mathtt{a}S\mathtt{b} \mid \mathtt{a}.$$

Example 8.3.7. Consider the language $\mathcal{L} = \{w : w \text{ contains at least three a's}\}$.
A CFG for this language is given by

$$\begin{aligned} S &\longrightarrow G\mathtt{a}G\mathtt{a}G\mathtt{a}G \\ G &\longrightarrow \mathtt{a}G \mid \mathtt{b}G \mid \varepsilon. \end{aligned}$$

Example 8.3.8. Consider the language $\mathcal{L} = \{w : w \text{ contains } \texttt{abb} \text{ as a substring}\}$.
A CFG for this language is given by

$$\begin{aligned} S &\longrightarrow G\texttt{abb}G \\ G &\longrightarrow \texttt{a}G \mid \texttt{b}G \mid \varepsilon. \end{aligned}$$

Example 8.3.9. Consider the language $\mathcal{L} = \{\texttt{a}^n\texttt{b}^{m+n}\texttt{a}^m : n, m \geq 0\}$.
A CFG for this language is given by

$$\begin{aligned} S &\longrightarrow LR \\ L &\longrightarrow \texttt{a}L\texttt{b} \mid \varepsilon \\ R &\longrightarrow \texttt{b}R\texttt{a} \mid \varepsilon. \end{aligned}$$

Example 8.3.10. Consider the alphabet $\Sigma = \{\texttt{a}, \texttt{b}, \varepsilon, \varnothing, (,), \cup, {}^*\}$ and the language $\mathcal{L}$ = all regular expressions.
A CFG for this language is given by

$$S \longrightarrow \varepsilon \mid \varnothing \mid \texttt{a} \mid \texttt{b} \mid (SS) \mid (S \cup S) \mid (S^*).$$

Example 8.3.11. Consider a programming language:

- identifiers = $(\texttt{a} \cup \cdots \cup \texttt{z})(\texttt{a} \cup \cdots \cup \texttt{z} \cup \texttt{0} \cup \cdots \cup \texttt{9})^*$
- numbers = positive integers
- statements of 2 types:
  - assignment
  - conditional
- program = a single statement

Now consider the alphabet $\Sigma = \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}, \texttt{0}, \texttt{1}, \ldots, \texttt{9}, >, =, <, \sqcup, \P\}$ and the language $\mathcal{L} =$ all valid programs.
A CFG for this language is given by

$$\begin{aligned} S &\longrightarrow A \mid C & & \\ D^+ &\longrightarrow \texttt{1} \mid \texttt{2} \mid \cdots \mid \texttt{9} & & \text{nonzero digit} \\ D &\longrightarrow \texttt{0} \mid D^+ & & \text{digit} \\ L &\longrightarrow \texttt{a} \mid \texttt{b} \mid \cdots \mid \texttt{z} & & \text{letter} \\ I &\longrightarrow ID \mid IL \mid L & & \text{identifier} \\ N &\longrightarrow ND \mid D^+ & & \text{number} \\ Q &\longrightarrow I \mid N & & \text{numeric quantity} \end{aligned}$$

$$\begin{aligned}
W &\to \text{␣} \mid \P \mid W && \text{whitespace}\\
W' &\to \varepsilon \mid W && \text{optional whitespace}\\
T &\to QW' = W'Q \mid QW' < W'Q \mid QW' > W'Q && \text{test}\\
A &\to IW' = W'Q && \text{assignment}\\
C &\to \texttt{if}\,WT\ \texttt{then}\,WS \mid \texttt{if}\,WTW\ \texttt{then}\,WSW\ \texttt{then}\,WS && \text{conditional.}
\end{aligned}$$

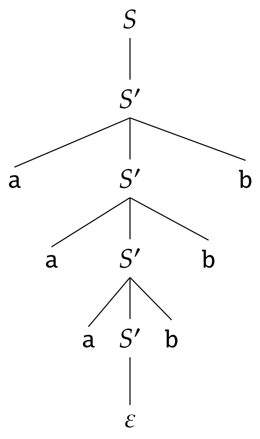
8.4 Ambiguity

Example 8.4.1. Consider the CFG with rules

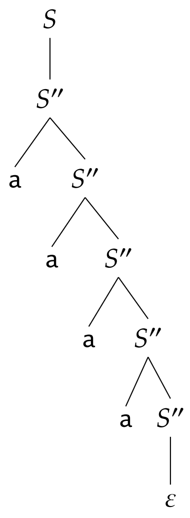
$$S \to S' \mid S''$$
$$S' \to \mathtt{a}S'\mathtt{b} \mid \varepsilon$$
$$S'' \to \mathtt{a}S'' \mid \varepsilon.$$

Note that this grammar generates the language $\{\mathtt{a}^n\mathtt{b}^n : n \geq 0\} \cup \mathtt{a}^*$.

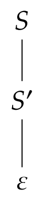
A parse tree for the string aaabbb is given by:

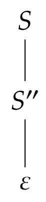


A parse tree for the string aaaa is given by:



We can find two parse trees for the string ε:





Definition 8.4.2

A grammar is called **ambiguous** if a string has more than one parse tree. Otherwise the grammar is called "unambiguous."

Example 8.4.3. Note that the CFG with rules

$$\begin{aligned} S &\to S' \mid S'' \\ S' &\to \mathtt{a}S'\mathtt{b} \mid \varepsilon \\ S'' &\to \mathtt{a}S'' \mid \varepsilon \end{aligned}$$

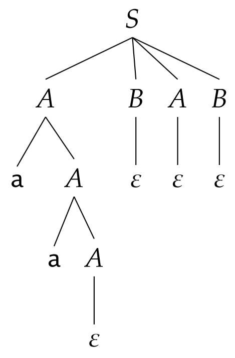
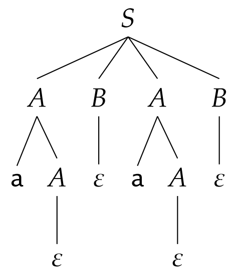
is ambiguous since the string ε has two parse trees. The ambiguity is fixed in the CFG with rules

$$\begin{aligned} S &\to S' \mid S'' \\ S' &\to \mathtt{a}S'\mathtt{b} \mid \varepsilon \\ S'' &\to \mathtt{a}S'' \mid \mathtt{a}. \end{aligned}$$

Example 8.4.4. Note that the CFG with rules

$$\begin{aligned} S &\to ABAB \\ A &\to \mathtt{a}A \mid \varepsilon \\ B &\to \mathtt{b}B \mid \varepsilon \end{aligned}$$

generates language $\mathtt{a}^*\mathtt{b}^*\mathtt{a}^*\mathtt{b}^*$ and ambiguously generates the string $\mathtt{aa}$:

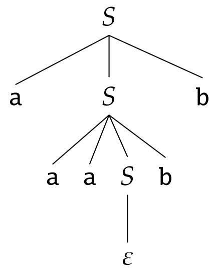
The ambiguity is fixed in the CFG with rules

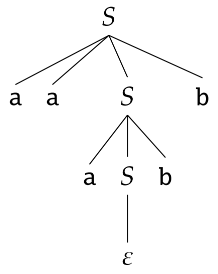
$$\begin{aligned} S &\to \overbrace{\varepsilon \mid A \mid B \mid AB \mid BA \mid ABA \mid BAB \mid ABAB}^{\text{8 disjoint languages}} \\ A &\to \mathtt{a}A \mid \mathtt{a} \\ B &\to \mathtt{b}B \mid \mathtt{b}. \end{aligned}$$

Example 8.4.5. Note that the CFG with a single rule

$$S \longrightarrow \mathtt{a}S\mathtt{b} \mid \mathtt{aa}S\mathtt{b} \mid \varepsilon$$

generates the language $\{\mathtt{a}^n\mathtt{b}^m : m \leq n \leq 2m\}$ and ambiguously generates the string aaabb:





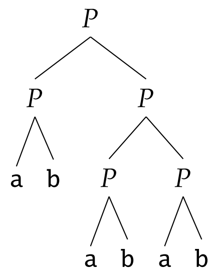
The ambiguity is fixed in the CFG with rules

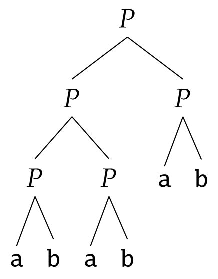
$$S \longrightarrow \mathtt{a}S\mathtt{b} \mid T \mid \varepsilon$$
$$T \longrightarrow \mathtt{aa}S\mathtt{b} \mid \mathtt{aab}.$$

Example 8.4.6. Note that the CFG with a single rule

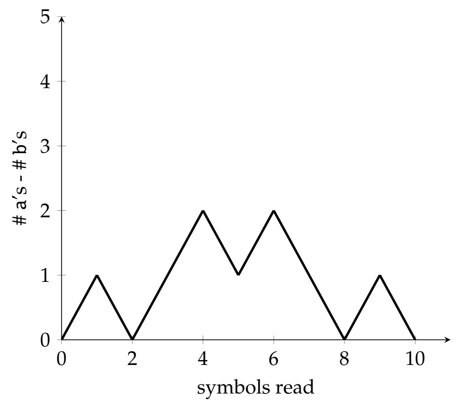
$$P \longrightarrow PP \mid \mathtt{a}P\mathtt{b} \mid \mathtt{ab}$$

generates the language of properly nested parentheses, where a represents (and b represents), and ambiguously generates the string ababab:

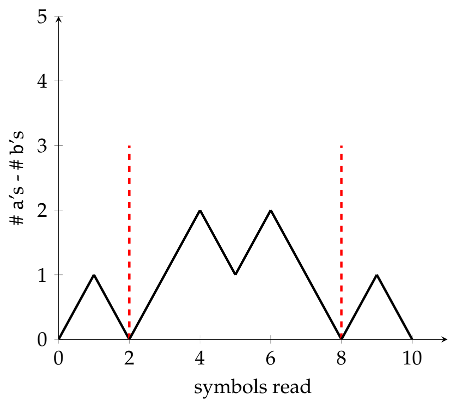




Consider the following graph:



Note that we can unambiguously split this graph into properly parenthesized expressions that have an outer pair of parentheses:

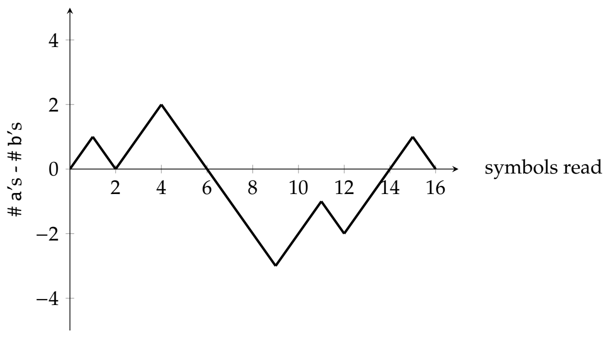


Then the ambiguity is fixed in the CFG with the single rule

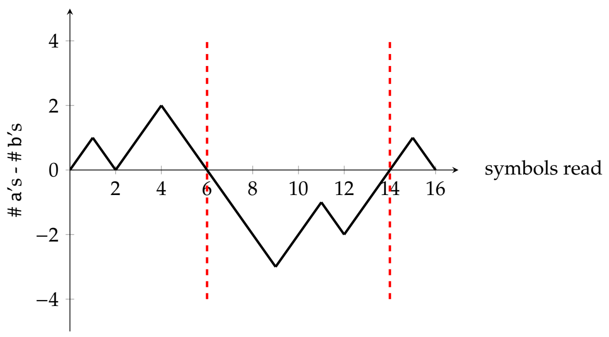
$$P \longrightarrow \underbrace{\texttt{ab} \mid \texttt{a}P\texttt{b}}_{\text{1 pair of outer parentheses}} \quad \mid \quad \underbrace{\texttt{ab}P \mid \texttt{a}P\texttt{b}P}_{\geq \text{ 2 pairs of outer parentheses}}$$

Example 8.4.7. Consider the language $\mathcal{L} = \{w : w \text{ has equally many } \texttt{a}\text{'s and } \texttt{b}\text{'s}\}$.

Consider the following graph:



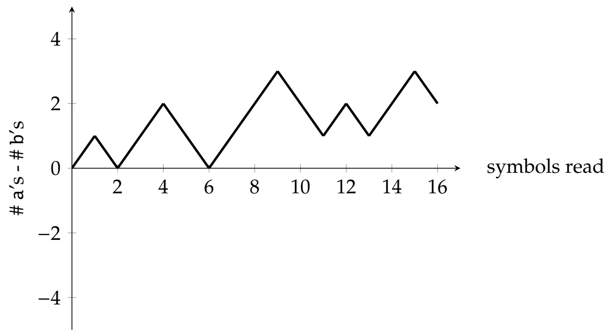
Note that we can unambiguously split this graph into maximal contiguous sections that lie completely above or below the x-axis:



Then the ambiguity is fixed in the CFG with rules

$$\begin{aligned} S &\longrightarrow S' \mid S'' \mid \varepsilon \\ S' &\longrightarrow P \mid PS'' \\ S'' &\longrightarrow Q \mid QS' \\ P &\longrightarrow \mathtt{ab} \mid \mathtt{a}P\mathtt{b} \mid \mathtt{ab}P \mid \mathtt{a}P\,\mathtt{b}P \\ Q &\longrightarrow \mathtt{ba} \mid \mathtt{b}Q\mathtt{a} \mid \mathtt{ba}Q \mid \mathtt{b}Q\mathtt{a}Q \end{aligned}$$

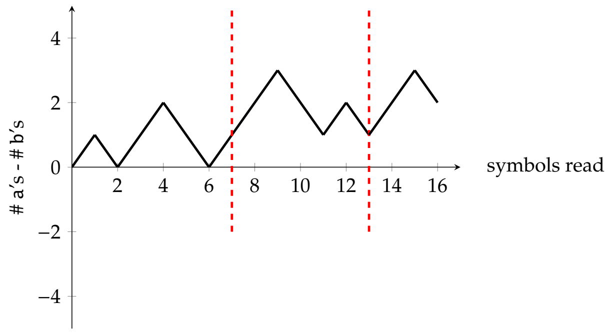
Example 8.4.8. Consider the language $\mathcal{L} = \{w : \text{every prefix of } w \text{ has at least as many a's as b's}\}$. Consider the following graph:



The two cases we can have are where

1. the balance starts at 0 and ends at 0, and where
2. the balance starts at 0 and ends up positive.

These two cases are disjoint:



Then the ambiguity is fixed in the CFG with a single rule

$$S \longrightarrow \underbrace{\varepsilon \mid P}_{\text{case 1}} \mid \underbrace{\mathtt{a}P \mid P\mathtt{a}S}_{\text{case 2}}.$$

8.5 Ambiguity in English

Consider the following sentences:

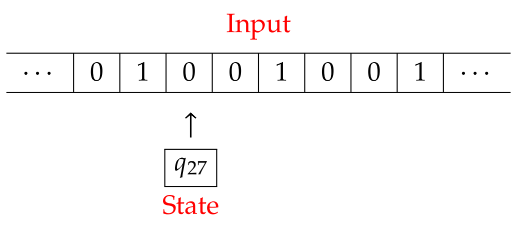
> The hunter attacked the bear with a stick.
>
> The complex houses married students.
>
> The bird the cat the dog the boy saw barked at chased flew away.
>
> Ship-shipping ships ship ship-shipping ships.

These examples drive home the point that English is very subtle and that you cannot represent it with a context-free grammar.

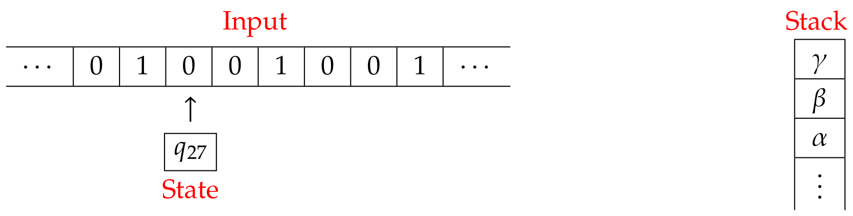
9 Pushdown Automata

9.1 Definitions

We may represent an NFA in the follwing way:

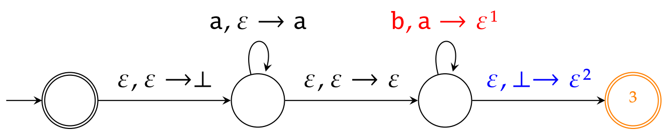


Then a PDA may be represented in the following way:



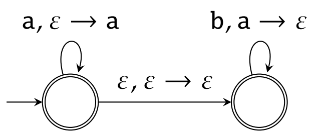
Transitions in PDAs are in the form of $\alpha, \beta \longrightarrow \gamma$ where α is the input symbol to read (read nothing if $\alpha = \varepsilon$), β is the symbol to pop from the top of the stack (do not pop if $\beta = \varepsilon$), and γ is the symbol to push onto the stack (do not push if $\gamma = \varepsilon$).

Example 9.1.1. A PDA for $\mathcal{L} = \{\mathtt{a}^n\mathtt{b}^n : n \geq 0\}$ is given by:

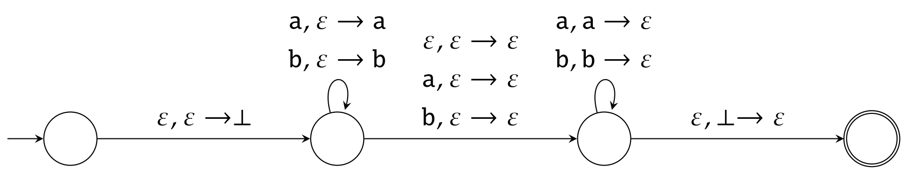

$\mathtt{a}, \varepsilon \longrightarrow \mathtt{a}$
$\mathtt{b}, \mathtt{a} \longrightarrow \varepsilon$[1]
$\varepsilon, \varepsilon \longrightarrow \perp$
$\varepsilon, \varepsilon \longrightarrow \varepsilon$
$\varepsilon, \perp \longrightarrow \varepsilon$[2]
3

Note that the input alphabet is $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ while the stack alphabet is $\Gamma = \{\perp, \mathtt{a}\}$ where $\perp$ is the bottom marker used to the bottom of the stack.

Example 9.1.2. A PDA for $\mathcal{L} = \{\mathtt{a}^n\mathtt{b}^m : n \geq m \geq 0\}$ is given by:

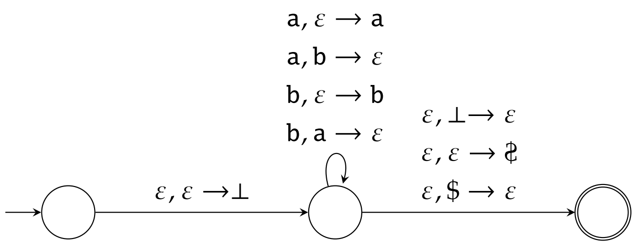

$\mathtt{a}, \varepsilon \longrightarrow \mathtt{a}$
$\mathtt{b}, \mathtt{a} \longrightarrow \varepsilon$
$\varepsilon, \varepsilon \longrightarrow \varepsilon$

Example 9.1.3. A PDA for $\mathcal{L} = \left\{w : w = w^R\right\}$ is given by:

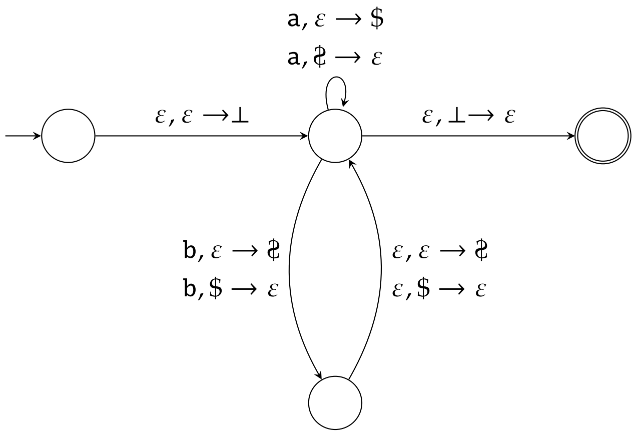

$\mathtt{a}, \varepsilon \longrightarrow \mathtt{a}$
$\mathtt{b}, \varepsilon \longrightarrow \mathtt{b}$
$\varepsilon, \varepsilon \longrightarrow \perp$
$\varepsilon, \varepsilon \longrightarrow \varepsilon$
$\mathtt{a}, \varepsilon \longrightarrow \varepsilon$
$\mathtt{b}, \varepsilon \longrightarrow \varepsilon$
$\mathtt{a}, \mathtt{a} \longrightarrow \varepsilon$
$\mathtt{b}, \mathtt{b} \longrightarrow \varepsilon$
$\varepsilon, \perp \longrightarrow \varepsilon$

[1] Usable only if next symbol is b and a is on top of the stack.
[2] Usable only if $\perp$ is on top of the stack.
[3] The PDA blows up if input is not finished.

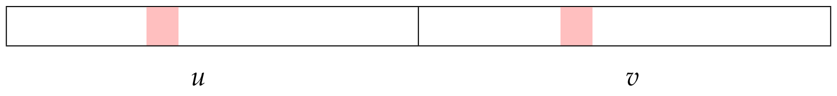
Example 9.1.4. A PDA for $\mathcal{L} = \{w : w \text{ has equally many } \mathtt{a}\text{'s and } \mathtt{b}\text{'s}\}$ is given by:



Example 9.1.5. A PDA for $\mathcal{L} = \{w : w \text{ has twice as many } \mathtt{a}\text{'s as } \mathtt{b}\text{'s}\}$ is given by:

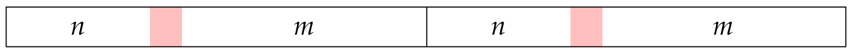


Example 9.1.6. Consider the language $\mathcal{L} = \{uv : |u| = |v| \wedge u \neq v\}$. A string $w \in \mathcal{L}$ may be visually represented by

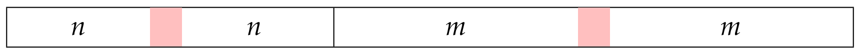


where the pink cells represent disagreement.

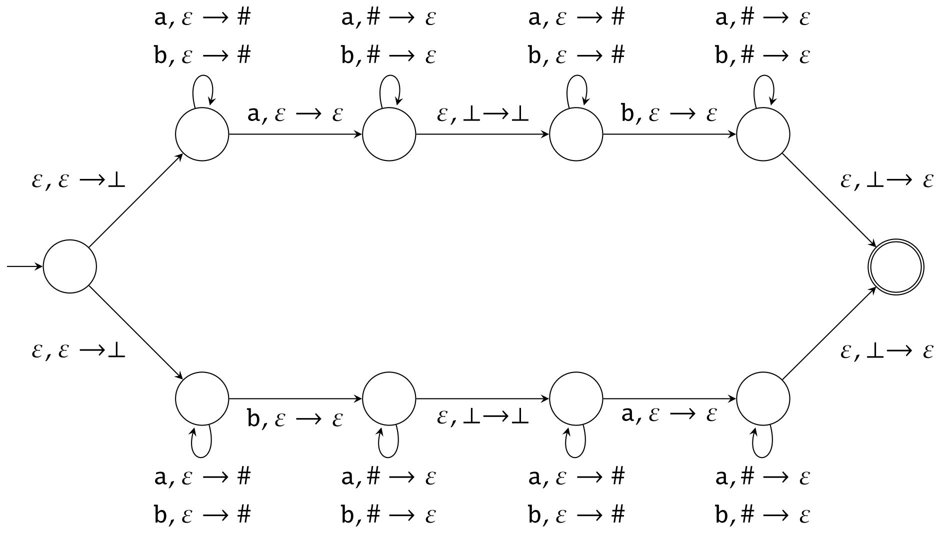
One may intuitively divide the string up like so:



However, it is very hard to find a PDA for this language from this. Dividing it up in a different way makes it a lot easier:



Then a PDA for $\mathcal{L}$ is given by:



10 Non-context free languages

10.1 Pumping lemma

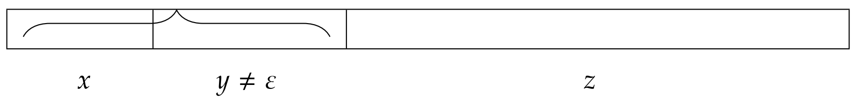
Definition 10.1.1

$\mathcal{L}$ is context free (CFL) if it is generated by a CFG.

Lemma 10.1.2: Pumping lemma for regular languages

Let $\mathcal{L}$ be regular then there exists $p \in \mathbb{N}$ ("pumping length") such that every $s \in \mathcal{L}$ of length $|s| \geq p$ can be decomposed as $s = xyz$ where

1. $|xy| \leq p$,
2. $|y| \neq 0$, and
3. $\forall\, i = 0, 1, 2, \cdots : xy^i z \in \mathcal{L}$.


x
$y \neq \varepsilon$
z