CS 181 (Introduction to Formal Languages and Automata Theory)

March 31, 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: $\{\mathtt{a}, \mathtt{b}, \dots, \mathtt{c}\}$

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. $\{\mathtt{acclaim}, \mathtt{aim}, \mathtt{brim}, \dots\}$

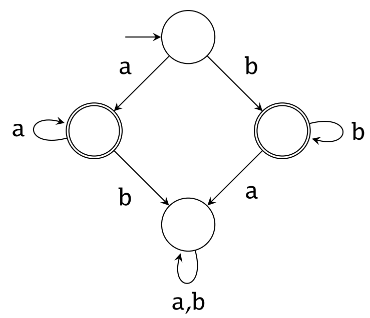
Example 1.1.12. $\{\mathtt{0}, \mathtt{1}, \mathtt{00}, \mathtt{11}, \dots\}$

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).



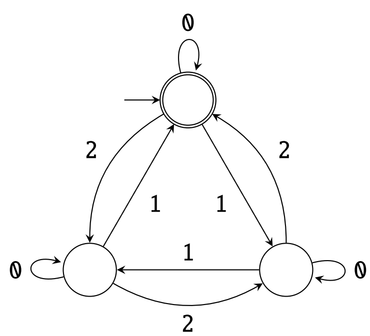
Input	Output
ε	reject
ab	reject
aaa	accept
bb	accept

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.

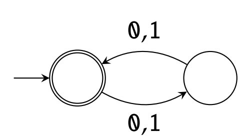
Example 1.2.2.



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

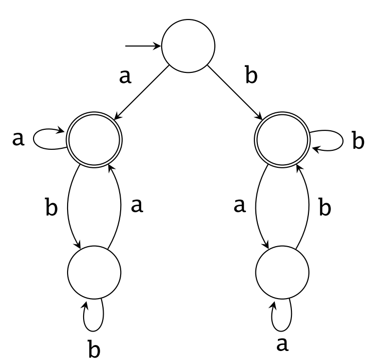
Alphabet: $\{\mathtt{0}, \mathtt{1}, \mathtt{2}\}$, language: $\{w : 3 \mid \sum w_i\}$

Example 1.2.3.



Alphabet: $\{\mathtt{0}, \mathtt{1}\}$, language: $\{w : 2 \mid |w|\}$

Example 1.2.4.

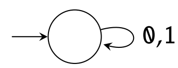


Alphabet: $\{\mathtt{a}, \mathtt{b}\}$, language: $\left\{w : w \neq \varepsilon \wedge w_1 = w_{|w|}\right\}$

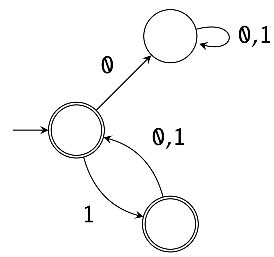
1.3 Designing DFAs

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

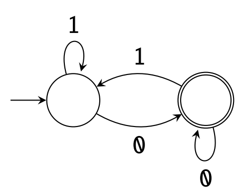
Example 1.3.1. Language: $\varnothing$



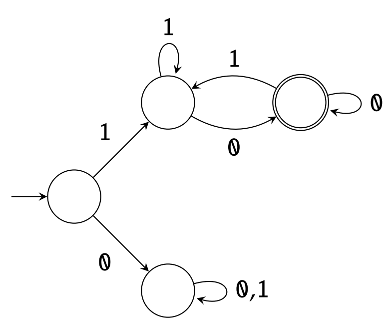
Example 1.3.2. Language: $\{w : \text{every odd position is a } \texttt{1}\}$



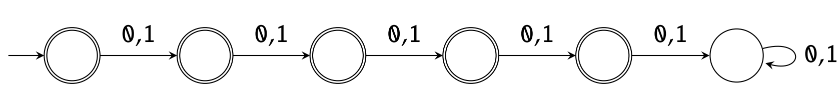
Example 1.3.3. Language: $\{w : w \text{ ends in } \texttt{0}\}$



Example 1.3.4. Language: $\{w : w \text{ begins with } \texttt{1}, \text{ ends with } \texttt{0}\}$



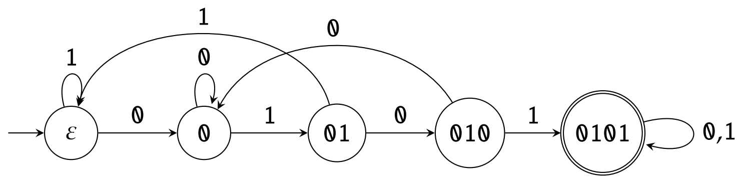
Example 1.3.5. Language: $\{w : |w| \leq 4\}$



Example 1.3.6. Language: $\{w : 1000 \mid |w|\}$

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

Example 1.3.7. Language: $\{w : w \text{ contains } \texttt{0101} \text{ as a substring}\}$



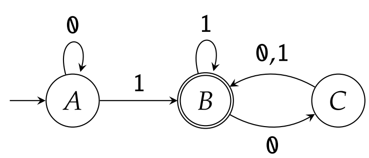
1.4 Formal definitions

Definition 1.4.1

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

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

Example 1.4.2.

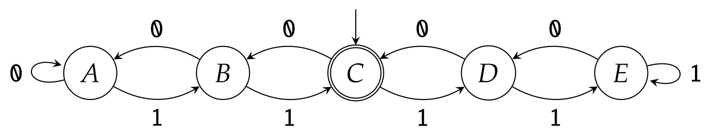


Formal description: $(\{A, B, C\}, \{\texttt{0}, \texttt{1}\}, \delta, A, \{B\})$ where δ is defined by the table

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

Example 1.4.3. Formal description: $(\{A, B, C, D, E\}, \{\texttt{0}, \texttt{1}\}, \delta, C, \{C\})$ where δ is defined by the table

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



Example 1.4.4. Formal description for Example 1.3.6: $(\{0, 1, 2, \ldots, 999\}, \{\mathtt{0}, \mathtt{1}\}, \delta, 0, \{0\})$ where $\delta(q, \sigma) = (q + 1) \mod 1000$.

Definition 1.4.5

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

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

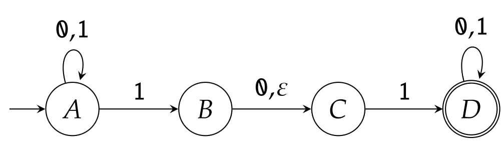
Note.

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

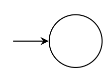
Example 2.1.3. Output table for Example 2.1.1:

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

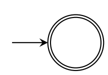
Language: all strings containing `101` or `11`

2.2 Using shortcuts

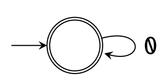
Example 2.2.1. Language: $\varnothing$



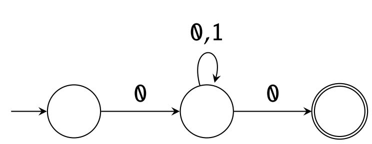
Example 2.2.2. Language: $\{\varepsilon\}$



Example 2.2.3. Language: $\{w : w$ doesn't contain `1`$\}$

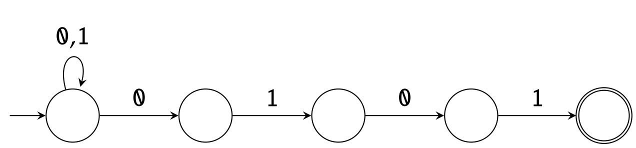


Example 2.2.4. Language: $\{w : |w| \geq 2$ and w starts and ends with `0`$\}$

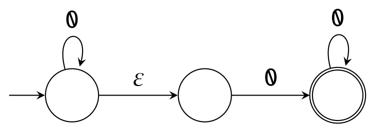


2.3 pattern matching

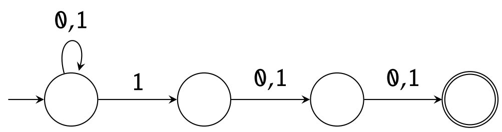
Example 2.3.1. Language: $\{w :$ conatins `0101`$\}$



Example 2.3.2. Language: $\left\{ w : w = \underbrace{\mathtt{00\ldots0}}_{\geq 0}\underbrace{\mathtt{11\ldots1}} \geq 0 \underbrace{\mathtt{00\ldots0}} \geq 1 \right\}$

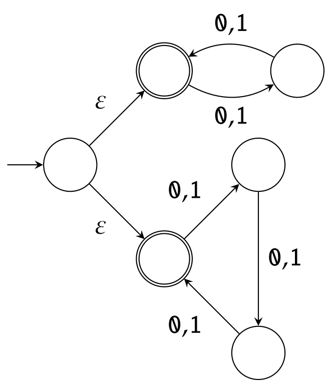


Example 2.3.3. Language: $\{w : w$ has a 1 in the 3rd position from the end$\}$



2.4 Alternatives

Example 2.4.1. Language: $\{w : 2 \mid |w| \vee 3 \mid |w|\}$



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