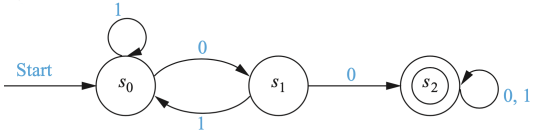
CSC 404 - Foundations of Computation

Section 1.1 – Finite Automaton

Finite Automaton - Example 1

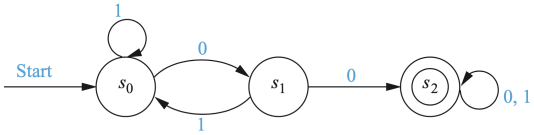


IDEA: GIVEN A BINARY STRING w, DO WE END AT s_2?

001: $s_0 \xrightarrow{0} s_1 \xrightarrow{0} s_2 \xrightarrow{1} s_2$ YES!

101: $s_0 \xrightarrow{1} s_0 \xrightarrow{0} s_1 \xrightarrow{1} s_0$ NO!

Finite Automaton - Example 1

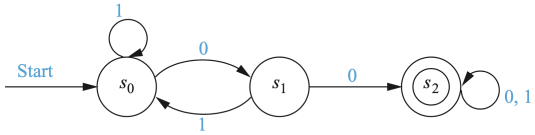


Example 1.1.1.

Determine whether each of these strings is recognized by the finite-state automaton above.

- 1001001 (73 in binary)
- 101010 (42 in binary)
- 100111001 (313 in binary)

Finite Automaton - Example 1

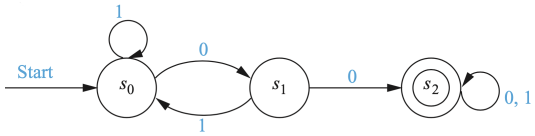


Example 1.1.1.

Determine whether each of these strings is recognized by the finite-state automaton above.

- 1001001 (73 in binary) YES!

$s_0 \xrightarrow{1} s_0 \xrightarrow{0} s_1 \xrightarrow{0} s_2 \xrightarrow{1} s_2 \xrightarrow{0} s_2 \xrightarrow{0} s_2 \xrightarrow{1} s_2$



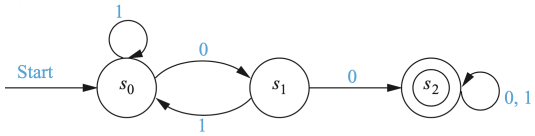
Example 1.1.1.

Determine whether each of these strings is recognized by the finite-state automaton above.

- 1001001 (73 in binary)
- 101010 (42 in binary) No!

$s_0 \xrightarrow{1} s_0 \xrightarrow{0} s_1 \xrightarrow{1} s_0 \xrightarrow{0} s_1 \xrightarrow{1} s_0 \xrightarrow{0} s_1 \Big\} \neq s_2$

Finite Automaton - Example 1



Example 1.1.1.

Determine whether each of these strings is recognized by the finite-state automaton above.

- 1001001 (73 in binary)
- 101010 (42 in binary)
- 100111001 (313 in binary) Yes! $s_0 \xrightarrow{1} s_0 \xrightarrow{0} s_1 \xrightarrow{0} s_2 \xrightarrow{1} s_2 \to \cdots \to s_2$

Definition 1.1.2.

An alphabet is a finite, nonempty set of symbols. Typically, we use the Greek letter Sigma, Σ, for an alphabet. Common examples of alphabets include:

1. $\Sigma = \{0, 1\}$, the binary alphabet.
2. $\Sigma = \{a, b, c, \ldots, y, z\}$, the set of all lower-case letters.
3. The set of all ASCII characters, or the set of all printable ASCII characters.

Definition 1.1.2.

An alphabet is a finite, nonempty set of symbols. Typically, we use the Greek letter Sigma, Σ, for an alphabet. Common examples of alphabets include:

1. $\Sigma = \{0, 1\}$, the binary alphabet.
2. $\Sigma = \{a, b, c, \ldots, y, z\}$, the set of all lower-case letters.
3. The set of all ASCII characters, or the set of all printable ASCII characters.

Definition 1.1.3.

A string (or sometimes word) is a finite sequence of symbols chosen from some alphabet.

e.g. * 101010 is a string/word over $\Sigma = \{0,1\}$ and $|101010| = 6$

* woocode is a string/word over $\Sigma = \{a, b, \ldots, z\}$ and $|woocode| = 7$

* print("woocode") is a word over ASCII char with length = 16

Definition 1.1.2.

An alphabet is a finite, nonempty set of symbols. Typically, we use the Greek letter Sigma, Σ, for an alphabet. Common examples of alphabets include:

1. $\Sigma = \{0, 1\}$, the binary alphabet.
2. $\Sigma = \{a, b, c, \ldots, y, z\}$, the set of all lower-case letters.
3. The set of all ASCII characters, or the set of all printable ASCII characters.

Definition 1.1.3.

A string (or sometimes word) is a finite sequence of symbols chosen from some alphabet.

Definition 1.1.4.

The length of a string is the number of positions for symbols in the string. The standard notation for the length of a string ω is $|\omega|$. For example, $|10101| = 5$ and $|111| = 3$.

Definition 1.1.2.

An alphabet is a finite, nonempty set of symbols. Typically, we use the Greek letter Sigma, Σ, for an alphabet. Common examples of alphabets include:

1. $\Sigma = \{0, 1\}$, the binary alphabet.
2. $\Sigma = \{a, b, c, \ldots, y, z\}$, the set of all lower-case letters.
3. The set of all ASCII characters, or the set of all printable ASCII characters.

Definition 1.1.3.

A string (or sometimes word) is a finite sequence of symbols chosen from some alphabet.

Definition 1.1.4.

The length of a string is the number of positions for symbols in the string. The standard notation for the length of a string ω is $|\omega|$. For example, $|10101| = 5$ and $|111| = 3$.

Definition 1.1.5.

The empty string is the string with zero occurrences of symbols. We typically denote this string by ϵ or $\emptyset$. Moreover, $|\epsilon| = 0$.

Definition 1.1.6.

The set of all strings over an alphabet Σ is denoted Σ^*.

Example 1.1.7.

If $\Sigma = \{0, 1\}$, then All Bit Strings

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, \cdots\}$$

Example 1.1.8.

If $\Sigma = \{a, b, c, \ldots, y, z\}$, then

$$\Sigma^* = \{\epsilon, a, b, c, \cdots, y, z, aa, ab, ac, \cdots, zy, zz, aaa, aab, \cdots\}$$

Definition 1.1.9.

If Σ is an alphabet, we define Σ^k to be the set of strings of length k, each of whose symbols are in Σ.

Example 1.1.10.

If $\Sigma = \{0, 1\}$, then

$$\Sigma^1 = \{0, 1\}$$

$$\Sigma^2 = \{00, 01, 10, 11\}$$

$$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

$$\Sigma^4 = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$$

* $\sum^4$ Gives all possible Farmer, Wolf, Goat, Cabbage states (valid and invalid)

Definition 1.1.9.

If Σ is an alphabet, we define Σ^k to be the set of strings of length k, each of whose symbols are in Σ.

Example 1.1.10.

If $\Sigma = \{0, 1\}$, then

$$\Sigma^1 = \{0, 1\}$$

$$\Sigma^2 = \{00, 01, 10, 11\}$$

$$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

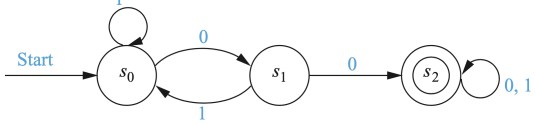
$$\Sigma^4 = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$$

Remark 1.1.11.

Another way to view Σ^* (set of all strings over an alphabet) is the following:

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots$$

$\Sigma = \{0,1\} \Rightarrow \Sigma^2 = \{00, 01, 10, 11\}$



Example 1.1.12.

Identify all strings of length 2 the finite automaton accepts (and rejects)

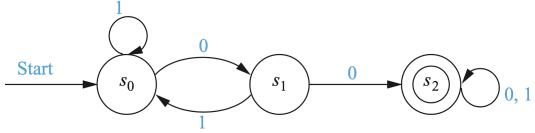
ACCEPT:

00

REJECT:

01 (end at s_0)
10 (end at s_1)
11 (end at s_0)

$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$



Example 1.1.13.

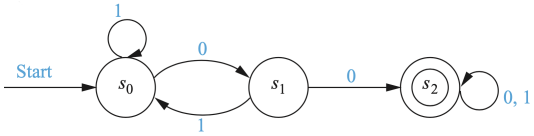
Identify all strings of length 3 the finite automaton accepts (and rejects)

ACCEPT:

000
001
100

REJECT:

010
011
101
110
111



Example 1.1.14.

Identify all strings of length 4 the finite automaton accepts (and rejects)

ACCEPT:

0000 1000
0001 1001
0010 1100
0011
0100

REJECT:

0101 1010
0110 1011
0111 1101
1110
1111

$\Sigma^4 = \{$ 0000, 0001, 0010, 0011,
0100, 0101, 0110, 0111,
1000, 1001, 1010, 1011,
1100, 1101, 1110, 1111 $\}$

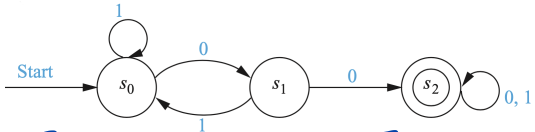
Definition 1.1.15.

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

1. Q is a finite set called the states,
2. Σ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma \to Q$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept (final) states. (MAY HAVE MULTIPLE FINAL STATES)

δ Needs 2 inputs! → current state (From Q)
→ character (From Σ)

Finite Automaton - Example 1



$Q = \{s_0, s_1, s_2\}$

$\Sigma = \{0, 1\}$

$q_0 = s_0$

$F = \{s_2\}$

WHAT ABOUT δ?

δ	0	1
s_0	s_1	s_0
s_1	s_2	s_0
s_2	s_2	s_2

Definition 1.1.16.

A set of strings all of which are chosen from some Σ^*, where Σ is a given alphabet, is called a language.

Remark 1.1.17.

The term 'language' is actually quite natural since common languages can be viewed as sets of strings. For example,

- The collection of legal English words is a set of strings over the alphabet that consists of all the letters.
- C (or any programming language) where the legal programs are a subset of the possible strings that can be formed from the alphabet of the language. Here the alphabet is a subset of the ASCII characters. (The exact alphabet differs amongst programming languages.)

Definition 1.1.16.

A set of strings all of which are chosen from some Σ^*, where Σ is a given alphabet, is called a language.

Remark 1.1.17.

The term 'language' is actually quite natural since common languages can be viewed as sets of strings. For example,

- The collection of legal English words is a set of strings over the alphabet that consists of all the letters.
- C (or any programming language) where the legal programs are a subset of the possible strings that can be formed from the alphabet of the language. Here the alphabet is a subset of the ASCII characters. (The exact alphabet differs amongst programming languages.)

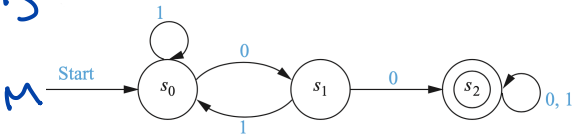
Definition 1.1.18.

If A is the set of all strings that machine M accepts, we say that A is the language of machine M and write $L(M) = A$. We say that M recognizes A or that M accepts A. (We typically say a machine will *accept* a particular string and that a machine will *recognize* a language)

Example 1.1.19.

Determine the set of all bit strings that the following finite-state automaton will recognize.

$\Sigma = \{0, 1\}$



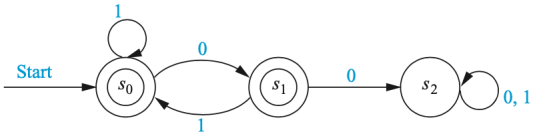
$L(M) = \{x \mid x \text{ contains } 00\}$

"Bit strings that contain 00"

Finite Automaton - Example 2

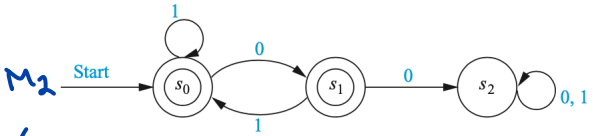
Example 1.1.20.

Determine the set of all bit strings that the following finite-state automaton will recognize.



Example 1.1.20.

Determine the set of all bit strings that the following finite-state automaton will recognize.



ACCEPT:

ϕ, 0, 1, 01, 10, 11, 010, 011, 101, 110, 111

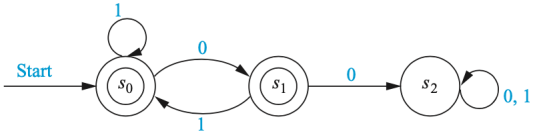
REJECT:

00, 000, 001, 100

$L(M_2) = \{x \mid x \text{ does not contain } 00\}$

Example 1.1.20.

Determine the set of all bit strings that the following finite-state automaton will recognize.



NOTE: $L(M_2) = \{x \mid x \text{ does not contain } 00\}$

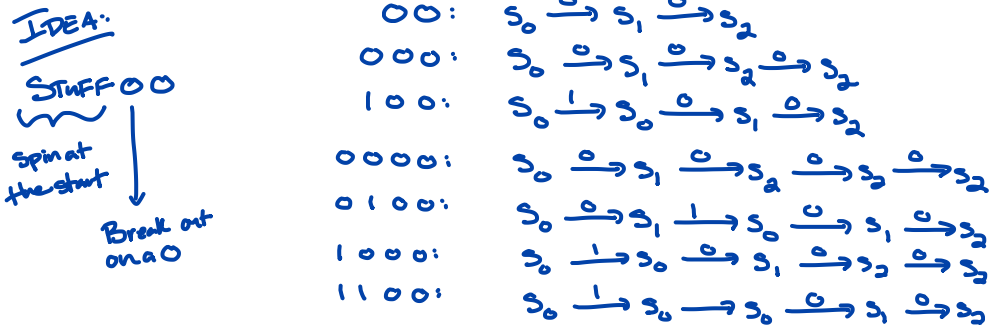
$= \sum^* - \{x \mid x \text{ contains } 00\}$

$= \sum^* - L(M_1)$

} In M_2 we "swapped" the states s_0, s_1, s_2.

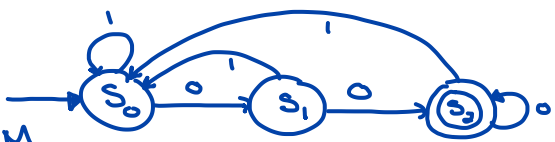
Example 1.1.21.

Construct a finite-state automaton that recognizes the set of all bit strings that ends with two 0s.



Example 1.1.21.

Construct a finite-state automaton that recognizes the set of all bit strings that ends with two 0s.



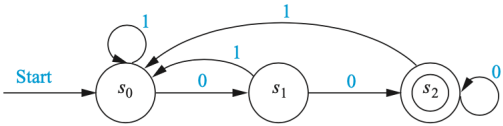
"If you see a 1, then go back to the start"

$L(M_3) = \{x \mid x \text{ ends with } 00\}$

Finite Automaton - Example 3

Example 1.1.21.

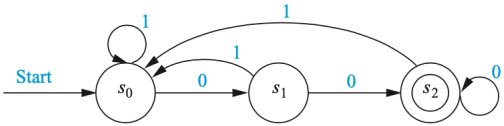
Construct a finite-state automaton that recognizes the set of all bit strings that ends with two 0s.



Finite Automaton - Example 3

Example 1.1.21.

Construct a finite-state automaton that recognizes the set of all bit strings that ends with two 0s.



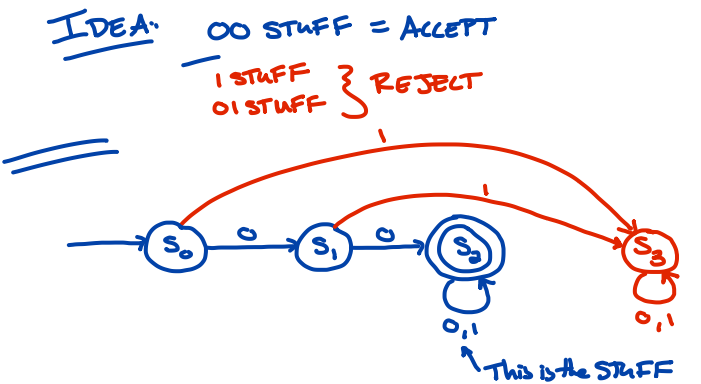
LATER: $\Sigma^* 00$

OR $\{0,1\}^* 00$

"Regular Expressions"

Example 1.1.22.

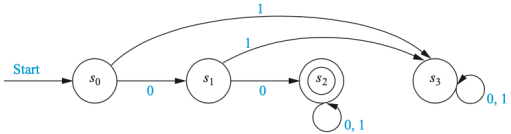
Construct a finite-state automaton that recognizes the set of all bit strings that begins with two 0s



Finite Automaton - Example 4

Example 1.1.22.

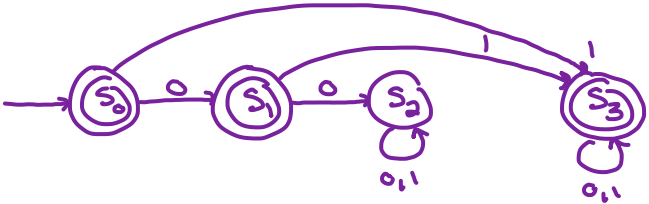
Construct a finite-state automaton that recognizes the set of all bit strings that begins with two 0s



Example 1.1.22.

Construct a finite-state automaton that recognizes the set of all bit strings that begins with two 0s

DO NOT



"Swap the states"

Finite Automaton - Example 5

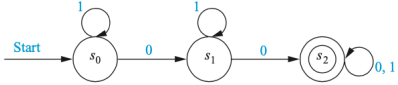
Example 1.1.23.

Construct a finite-state automaton that recognizes the set of all bit strings that contain two 0s.

Finite Automaton - Example 5

Example 1.1.23.

Construct a finite-state automaton that recognizes the set of all bit strings that contain two 0s.



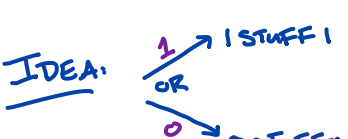
Finite Automaton - Example 6

Example 1.1.24.

Construct a finite-state automaton that recognizes the set of all bit strings that either start and end with a 0 or start and end with a 1.

Example 1.1.24.

Construct a finite-state automaton that recognizes the set of all bit strings that either start and end with a 0 or start and end with a 1.



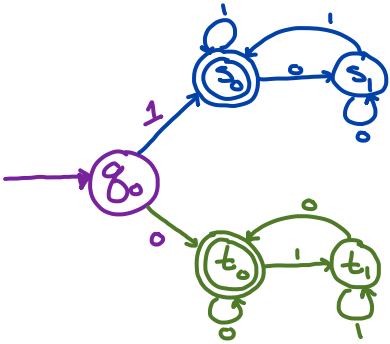
START AND END WITH 1 (1 STUFF 1)

e.g. 1, 11, 101, 111, 1001, 1011, 1101, 1111, ...

Finite Automaton - Example 6

Example 1.1.24.

Construct a finite-state automaton that recognizes the set of all bit strings that either start and end with a 0 or start and end with a 1.



Finite Automaton - Example 7

Example 1.1.25.

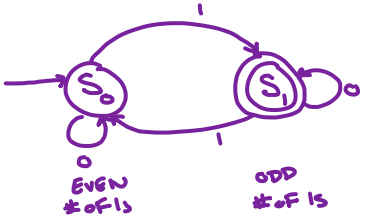
Construct a finite-state automaton that recognizes the set of all bit strings that contain an odd number of 1s and ending with at least two 0s.

Example 1.1.25.

Construct a finite-state automaton that recognizes the set of all bit strings that contain an odd number of 1s and ending with at least two 0s.

IDEA: TEST FOR ODD # OF 1s THEN TEST FOR END WITH 00.

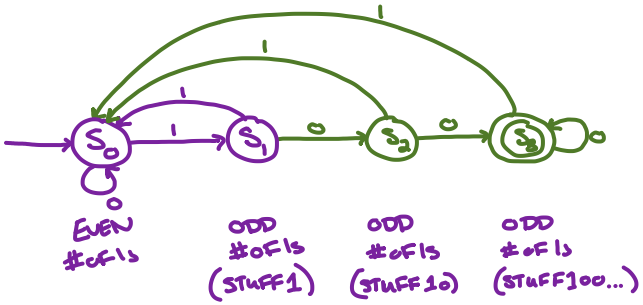
ODD # OF 1s (e.g. 1, 10, 01, 001, 010, 100, 0001, 0010, 0100, 1000, 0111, 1011, 1101, 1110, ...



Finite Automaton - Example 7

Example 1.1.25.

Construct a finite-state automaton that recognizes the set of all bit strings that contain an odd number of 1s and ending with at least two 0s.



Finite Automaton - Example 8

Example 1.1.26.

Construct a finite-state automaton that recognizes the set of all bit strings of zero or more 0 bits followed by a 1.

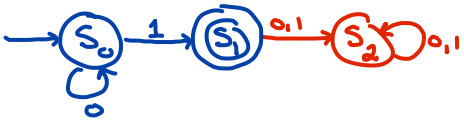
Finite Automaton - Example 8

Example 1.1.26.

Construct a finite-state automaton that recognizes the set of all bit strings of zero or more 0 bits followed by a 1.

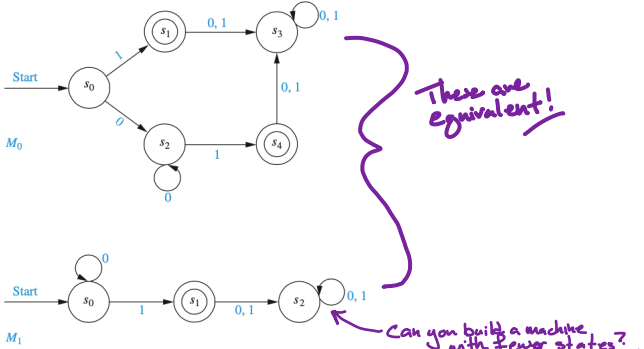
IDEA: 0...01 → ACCEPT

1STUFF
0...01STUFF } REJECT
0....0



Example 1.1.26.

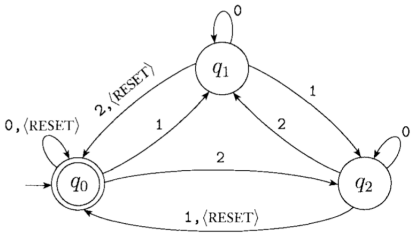
Construct a finite-state automaton that recognizes the set of all bit strings of zero or more 0 bits followed by a 1.



Finite Automaton - Example 9

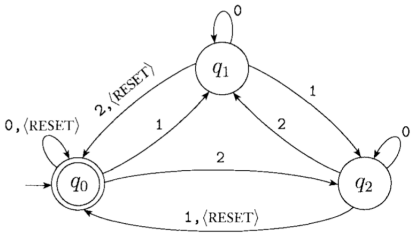
Example 1.1.27.

Consider the following three-state machine, which has a four-symbol input alphabet, $\Sigma = \{\langle RESET \rangle, 0, 1, 2\}$. (Here we treat $\langle RESET \rangle$ as a single symbol.)



Example 1.1.27.

Consider the following three-state machine, which has a four-symbol input alphabet, $\Sigma = \{\langle RESET \rangle, 0, 1, 2\}$. (Here we treat $\langle RESET \rangle$ as a single symbol.)



δ	0	1	2	⟨RESET⟩
q_0	q_0	q_1	q_2	q_0
q_1	q_1	q_2	q_0	q_0
q_2	q_2	q_0	q_1	q_0

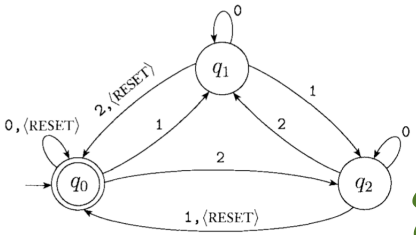
Example 1.1.27.

Consider the following three-state machine, which has a four-symbol input alphabet, $\Sigma = \{\langle RESET \rangle, 0, 1, 2\}$. (Here we treat $\langle RESET \rangle$ as a single symbol.)

+	0	1	2
0	0+0=0	0+1=1	0+2=2
1	1+0=1	1+1=2	1+2=3≡0 mod3
2	2+0=2	2+1≡0	2+2=4≡1 mod3

Example 1.1.27.

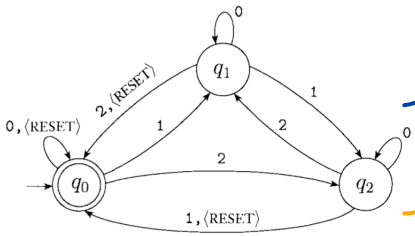
Consider the following three-state machine, which has a four-symbol input alphabet, $\Sigma = \{\langle RESET \rangle, 0, 1, 2\}$. (Here we treat $\langle RESET \rangle$ as a single symbol.)



$$\begin{cases} \delta(q_i, \langle RESET \rangle) = q_0 \quad \text{for all } i \\ \delta(q_i, j) = q_{(i+j \bmod 3)} \end{cases}$$

Example 1.1.27.

Consider the following three-state machine, which has a four-symbol input alphabet, $\Sigma = \{\langle RESET \rangle, 0, 1, 2\}$. (Here we treat $\langle RESET \rangle$ as a single symbol.)



1221: $q_0 \to q_2 \to q_1 \to q_0$

$1+2+2+1 \equiv 3+3 \equiv 6 \equiv 0 \pmod 3$

222: $q_0 \to q_2 \to q_1 \to q_0$

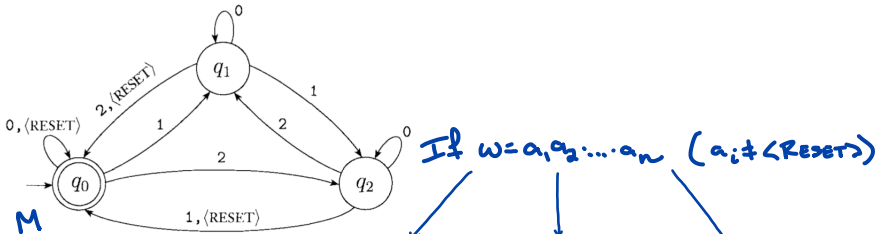
$2+2+2 \equiv 6 \equiv 0 \pmod 3$

1201: $q_0 \xrightarrow{1} q_1 \xrightarrow{2} q_0 \xrightarrow{1} q_1$

$1+2+0+1 = 4 \equiv 1 \pmod 3$

Example 1.1.27.

Consider the following three-state machine, which has a four-symbol input alphabet, $\Sigma = \{\langle RESET \rangle, 0, 1, 2\}$. (Here we treat $\langle RESET \rangle$ as a single symbol.)



Finite Automaton - Example 9

Example 1.1.27.

Consider the following three-state machine, which has a four-symbol input alphabet, $\Sigma = \{\langle RESET \rangle, 0, 1, 2\}$. (Here we treat $\langle RESET \rangle$ as a single symbol.)

Definition 1.1.28.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 w_3 \dots w_n$ be a string over the alphabet Σ. Then M **accepts** w if a sequence of states $r_0, r_1, \dots, r_n$ in Q exists with three conditions:

1. $r_0 = q_0$ (Start at the initial state)
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ (The machine goes from state-to-state correctly)
3. $r_n \in F$ (We end up in an accept/final state – the machine accepts the input)

Definition 1.1.29.

We say that M recognizes language A if $A = \{w \mid M \text{ accepts } w\}$

Definition 1.1.30.

A language is called a **regular language** if some finite automaton recognizes it.