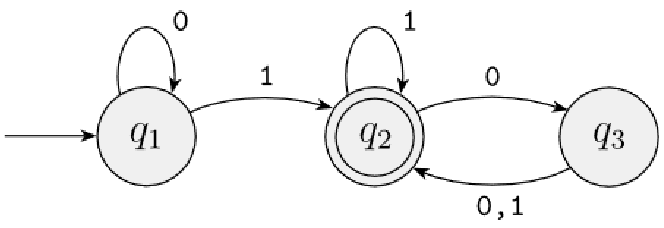
Ch 1.1 Finite Automata (S)

Markov chains ?

State diagram



M_1

- 3 states, labeled q_1, q_2, q_3, output *accept/reject*
- **start state**: q_1 (indicated by the arrow from nowhere)
- **accept state/final states**: q_2 (double circle)
- **transitions**: arrows

▼ Ex. 1101

1. start in state q_1
2. Read 1, follow transition from q_1 to q_2
3. Read 1, follow transition from q_2 to q_2
4. Read 0, follow transition from q_2 to q_3
5. Read 1, follow transition from q_3 to q_2
6. *Accept* because M_1 is in an accept state q_2 at the end

Formal Definition of a Finite Automaton

A *finite automaton* is a 5-tuple $(Q, \sum, \delta, q_0, F)$, where
1. Q is a finite set called the **states**,
2. $\sum$ is a finite set called the **alphabet**,
3. $\delta : Q \times \sum \longrightarrow Q$ is the **transition function**,
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the **set of accept states**.

▼ Example M_1

We can describe M_1 formally by writing $M_1 = (Q, \sum, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$,
2. $\sum = \{0, 1\}$,
3. δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

4. q_1 is the start state, and
5. $F = \{q_2\}$.

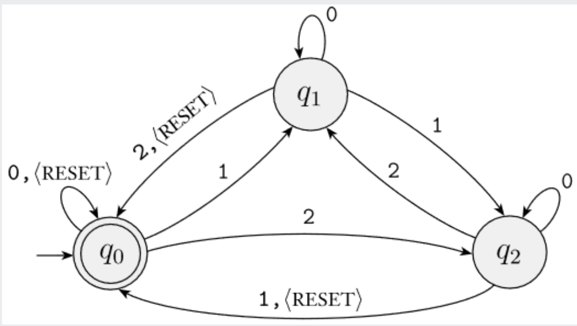
$A = \{w | w$ contains at least one 1 and an even number of 0s follow the last 1$\}$.

$L(M_1) = A$, or equivalently, M_1 recognizes A.

> Let A be the set of all strings M accepts, A is the language of machine M and write $L(M) = A$. We say that M recognizes A.

Examples of Finite Automata

▼ Example 1.13: Modulo 3



*view <RESET> as a single symbol

$\sum = \{<\text{RESET}>, 0, 1, 2\}$

This machine is running count of the sum of the numerical input symbols it reads, modulo 3.

▼ Example 1.15: no diagram but description

Consider a generalization of Example 1.13, using the same four-symbol alphabet $\sum$. For each $i \geq 1$ let A_i be the language of all strings where the sum of the numbers is a multiple of i, except that the sum is reset to 0 whenever the symbol <RESET> appears. For each A_i we give a finite automaton $B_i = (Q_i, \sum, \delta_i, q_0, \{q_0\})$, where Q_i is the set of i states $\{q_0, q_1, q_2, \ldots, q_{i-1}\}$, and we design the transition function δ_i so that for each j, if B_i is in q_j, the running sum is j, modulo i. For each q_j let

$\delta_i(q_j, 0) = q_j$,

$\delta_i(q_j, 1) = q_k$, where $k = j + 1 \mod i$,

$\delta_i(q_j, 2) = q_k$, where $k = j + 2 \mod i$, and

$\delta_i(q_j, <\text{RESET}>) = q_0$,

Formal Definition of Computation

Let $M = (Q, \sum, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 \ldots w_n$ be a string where each w_i is a member of the alphabet $\sum$. Then M **accepts** w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \ldots, n-1$, and
3. $r_n \in F$.

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

Designing Finite Automata

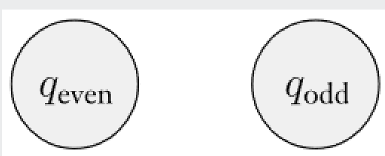
- put *yourself* in the place of the machine
- only remember certain crucial input

▼ All strings with an odd number of 1s.

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

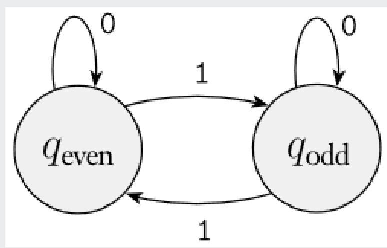
Possibilities

1. even so far, and
2. odd so far



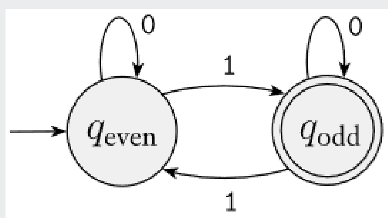
▼

Assign transitions (seeing how to go from one possibility to another)



set start state → the possibility having seem 0 symbols so far (ϵ) → q_{even} ∵ 0 is even

set accept states → q_{odd}



The Regular Operations

Let A and B be languages. We define the regular operations **union, concatenation,** and **star** as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- **Concatenation:** $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
- **Star**: $A^* = \{x_1x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

▼ Example 1.24

Let the alphabet $\sum$ be the standard 26 letters $\{a, b, \ldots, z\}$. If $A = \{\text{good, bad}\}$ and $B = \{\text{boy, girl}\}$, then

$A \cup B = \{\text{good, bad, boy, girl}\}$,

$A \circ B = \{\text{goodboy, goodgirl, badboy, badgirl}\}$, and

$A^* = \{\epsilon$, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, goodbadgood, goodbadbad, $\ldots\}$

The class of regular languages is closed under the union operation.
In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

The class of regular languages is closed under the concatenation operation.
In other words, if A_1 and A_2 are regular languages, so is $A_1 \circ A_2$.