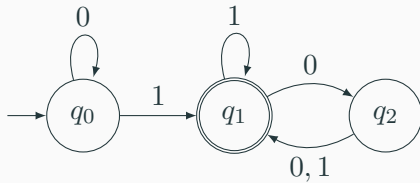


# Finite Automata

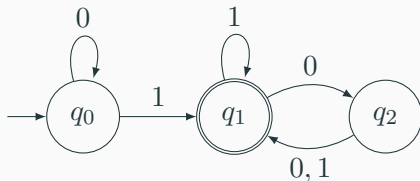
---

ian.mcloughlin@gmit.ie

## Finite Automaton: Example 1



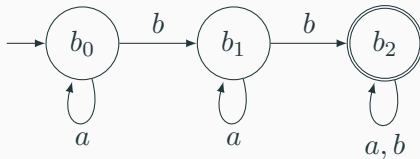
## Finite Automaton: Example 1



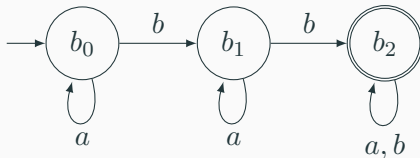
Try running the automaton on the following strings.

1101, 1, 01, 11, 0101010101, 100, 0100,  
110000, 0101000000, 0, 10, 101000

## Finite Automaton: Example 2



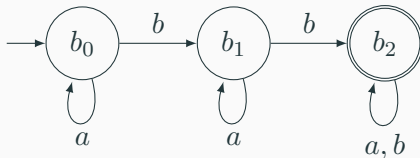
## Finite Automaton: Example 2



Try running the automaton on the following strings.

*aaaa, ababa, bababb, abaa*

## Finite Automaton: Example 2

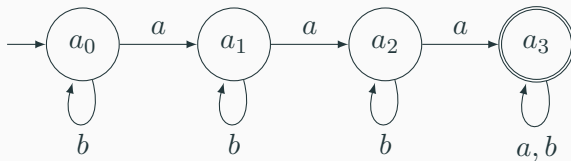


Try running the automaton on the following strings.

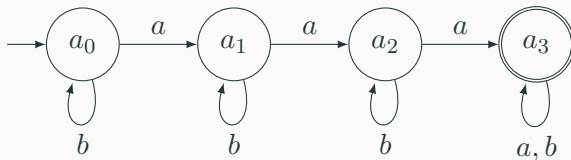
*aaaa, ababa, bababb, abaa*

Describe the strings that the automaton recognises.

## Finite Automaton: Example 3



## Finite Automaton: Example 3

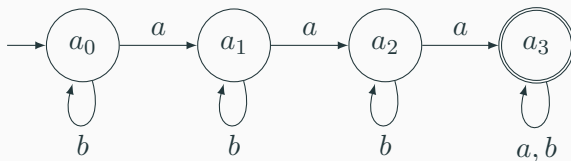


Try running the automaton on the following strings.

*aaaa, ababa, bababb, abaa*



## Finite Automaton: Example 3

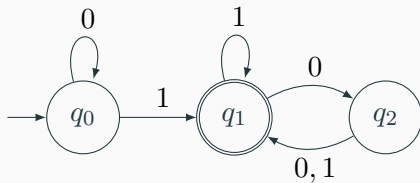


Try running the automaton on the following strings.

*aaaa, ababa, bababb, abaa*

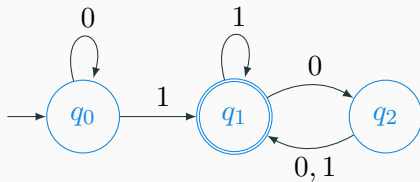
Describe the strings that the automaton recognises.

# Finite Automaton: Concepts



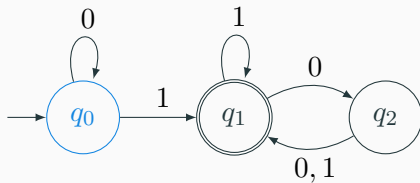
What are the essential concepts?

# Finite Automaton: Concepts



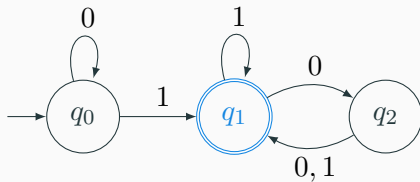
Set of states:  $Q = \{q_0, q_1, q_2\}$

# Finite Automaton: Concepts



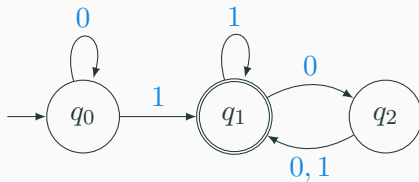
Initial state:  $q_0 \in Q$

# Finite Automaton: Concepts



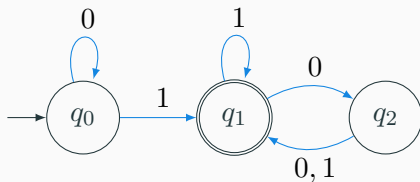
Set of final states:  $F = \{q_1\} \subseteq Q$

# Finite Automaton: Concepts



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

# Finite Automaton: Concepts



Transition function:  $\delta = \{((q_0, 0), q_0), ((q_0, 1), q_1), ((q_1, 0), q_2), \dots\}$

# Deterministic Finite Automaton (DFA) definition

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

$Q$  is a finite set of *states*,

$\Sigma$  is a finite set called the *alphabet*,

$\delta$  is the *transition function*  $(Q \times \Sigma \rightarrow Q)$ ,

$q_0$  is the *start state* ( $\in Q$ ), and

$F$  is the set of *accept states* ( $\subseteq Q$ ).



## Example 1 definition

$$Q = \{q_0, q_1, q_2\}$$

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

$$\delta = \{((q_0, 0), q_0), ((q_0, 1), q_1), ((q_1, 0), q_2), ((q_1, 1), q_1), \\ ((q_2, 0), q_1), ((q_2, 1), q_1)\}$$

$$q_0 = q_0$$

$$F = \{q_1\}$$

## Example 2 definition

$$Q = \{b_0, b_1, b_2\}$$

$$\Sigma = \{a, b\}$$

$$\delta = \{((b_0, a), b_0), ((b_0, b), b_1), ((b_1, a), b_1), ((b_1, b), b_2), \\ ((b_2, a), b_2), ((b_2, b), b_2)\}$$

$$q_0 = b_0$$

$$F = \{b_2\}$$

## Example 3 definition

$$Q = \{a_0, a_1, a_2, a_3\}$$

$$\Sigma = \{a, b\}$$

$$\delta = \{((a_0, a), a_1), ((a_0, b), a_0), ((a_1, a), a_2), ((a_1, b), a_1), \\ ((a_2, a), a_3), ((a_2, b), a_2)\}, ((a_3, a), a_3), ((a_3, b), a_3)\}$$

$$q_0 = a_0$$

$$F = \{a_3\}$$

# Non-determinism

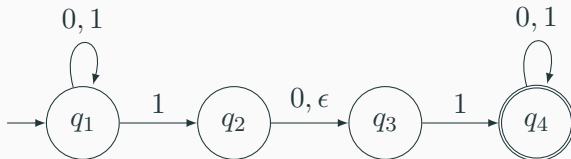
**DFAs** always have exactly one state to transition to when in any given state and reading any given symbol.

**Non-deterministic** finite automata can have any number of arrows for each state and symbol.

**The empty string  $\epsilon$**  is also used to label arrows that are followed without reading a character from the input, while also remaining in the original state.

**Non-determinism** can simplify automata but it can be shown that NFAs and DFAs recognise the same set of languages.

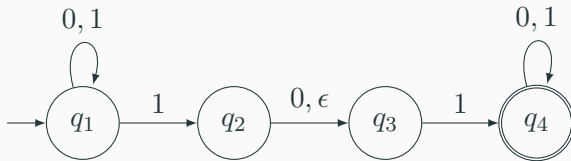
## NFA example



Try running the following strings on the automaton.

111101, 00001010, 1110,  $\epsilon$

## NFA example

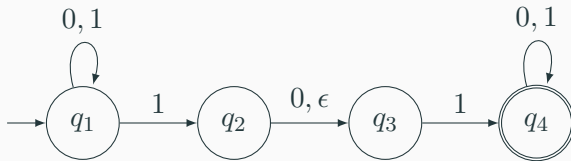


Try running the following strings on the automaton.

111101, 00001010, 1110,  $\epsilon$

Describe in words the strings that the automaton recognises.

## NFA example



Try running the following strings on the automaton.

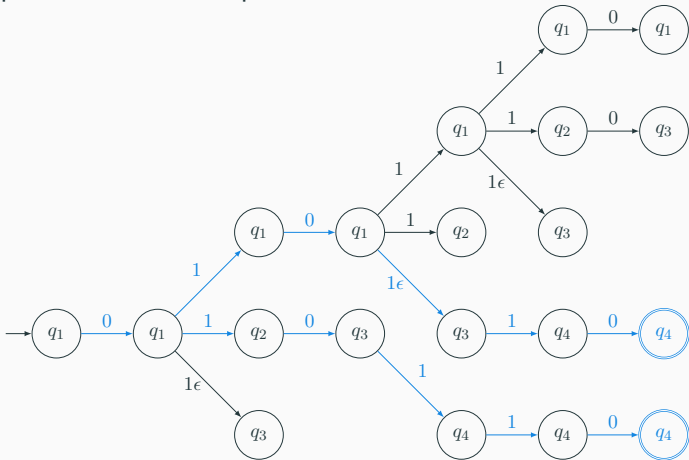
111101, 00001010, 1110,  $\epsilon$

Describe in words the strings that the automaton recognises.

(Answer: all strings that contain either 11 or 101.)

## NFA example

Example: 010110 on the previous NFA.



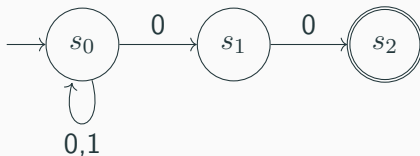


## NFA example

Construct an NFA with alphabet  $\{0, 1\}$  to recognise the language  $\{w \mid w \text{ ends with } 00\}$ . Try to do it with only three states.

## NFA example

Construct an NFA with alphabet  $\{0, 1\}$  to recognise the language  $\{w \mid w \text{ ends with } 00\}$ . Try to do it with only three states.



# Non-deterministic Finite Automaton (NFA) definition

An NFA is a 5-tuple  $(Q, \Sigma, \delta, q_0, F)$  where

$Q$  is a finite set of *states*,

$\Sigma$  is a finite set called the *alphabet*,

$\delta$  is the *transition function*  $(Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q))$ ,

$q_0$  is the *start state* ( $\in Q$ ), and

$F$  is the set of *accept states* ( $\subseteq Q$ ).

By  $\Sigma_\epsilon$  we mean  $\Sigma \cup \{\epsilon\}$ . e.g. When  $\Sigma = \{0, 1\}$ ,  $\Sigma_\epsilon = \{\epsilon, 0, 1\}$ .

## Powerset example

Take any set, say  $A = \{0, 1, 2\}$ . Its powerset is the set of all its subsets, and is denoted  $\mathcal{P}(A)$ .

$$\mathcal{P}(A) = \left\{ \{\}, \{0\}, \{1\}, \{2\}, \{0, 1\}, \{0, 2\}, \{1, 2\}, \{0, 1, 2\} \right\}$$

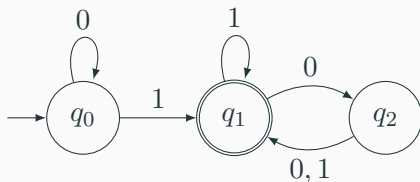
## Strings of length $p$

**Finite** means that the number of states is finite.

**Let**  $p$  be the number of states in an automaton.

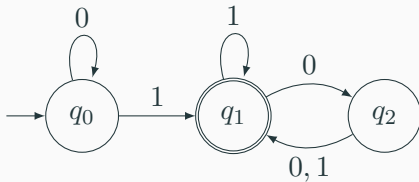
**Every** string of length at least  $p$  must visit one state twice.

Try the strings 000, 001, 010, etc on the following automaton.



## Example of looping a string

Consider the string 110001 on the following automaton.



1	1	0	0	0	0	0	0	0	1
$x$		$y$		$y$		$y$		$y$	$z$

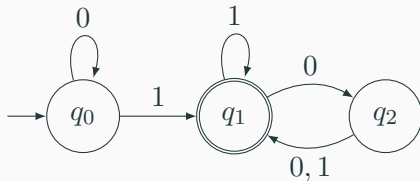
## Theorem

*Let  $A$  be a regular language. There is a positive integer  $p$  such that every string  $s$  of length at least  $p$  in  $A$  may be broken into three substrings  $s = xyz$  where:*

- *$xy^iz$  is in  $A$  for all non-negative integers  $i$ .*
- *The length of  $y$  is greater than zero ( $|y| > 0$ ).*
- *The length of  $xy$  is less than or equal to  $p$  ( $|xy| \leq p$ ).*

## Pumping lemma example

Consider the string 110001 on the following automaton.

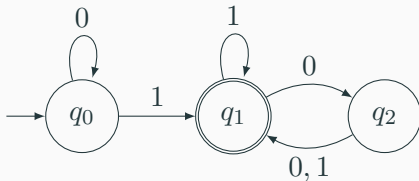


Note that  $|xy|$  must be less than  $p$ .



## Pumping lemma example

Consider the string 110001 on the following automaton.



Note that  $|xy|$  must be less than  $p$ .

1	1	1	1	1	1	1	1	0	0	0	1
$x$	$y$	$y$	$y$	$y$	$y$	$y$	$y$	$z$			

## No automaton recognises $\{0^i1^i\}$

Is there a finite automaton that recognises  $\{0^i1^i \mid i \in \mathbb{N}\}$ ?

If so, it has a finite number of states — let that number be  $p$ .

The string  $0^p1^p$  ( $p$  0's followed by  $p$  1's) must be accepted by the automaton.

By the pumping lemma, it can be broken into  $xyz$  where  $xy^iz$  is also accepted for all  $i \in \mathbb{N}$ ,  $y$  is of length greater than zero and  $xy$  is no longer than  $p$ .

So,  $|xy| \leq p$  and  $|y| > 0$ , meaning  $y$  must be a string of 0's.

However, then  $xyyz$  contains more 0's than 1's — a contradiction.