

# Computational Models — Lecture 2<sup>1</sup>

## Handout Mode

Iftach Haitner.

Tel Aviv University.

November 07, 2016

---

<sup>1</sup>Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.

## Computational Models - Lecture 2

- ▶ Non-Deterministic Finite Automata (NFA)
- ▶ Closure of Regular Languages Under  $\cup$ ,  $\parallel$ ,  $*$
- ▶ Regular **expressions**
- ▶ Equivalence with finite automata
  
- ▶ Sipser's book, [1.1 – 1.3](#)

# Part I

## Non-Deterministic Finite Automata

## DFA – formal definition, (reminder)

### Definition 1 (DFA)

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

- ▶  $Q$  is a finite set called the **states**,
- ▶  $\Sigma$  is a finite set called the **alphabet**,
- ▶  $\delta : Q \times \Sigma \mapsto Q$  is the **transition function**,
- ▶  $q_0 \in Q$  is the **start state**, and
- ▶  $F \subseteq Q$  is the set of **accept states**.

## Formal model of computation, (reminder)

### Definition 2

$M = (Q, \Sigma, \delta, q_0, F)$  **accepts**  $w \in \Sigma^*$  if  $\hat{\delta}(q_0, w) \in F$ .

### Definition 3 ( $\hat{\delta}$ )

For DFA  $M = (Q, \Sigma, \delta, q_0, F)$ , define  $\hat{\delta}: Q \times \Sigma^* \mapsto Q$  by

$$\hat{\delta}(q, w) = \begin{cases} \delta(\hat{\delta}(q, w_1, \dots, w_{n-1}), w_n), & n = |w| \geq 1 \\ q, & w = \varepsilon. \end{cases}$$

## The language of a DFA, (reminder)

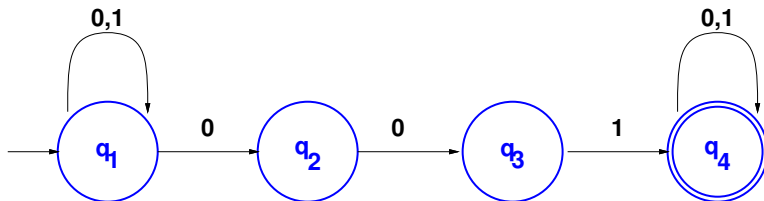
### Definition 4

The language of a DFA  $M$ , denoted  $\mathcal{L}(M)$ , is the set of strings that  $M$  accepts.

### Definition 5

A language is called regular, if some deterministic finite automaton accepts it.

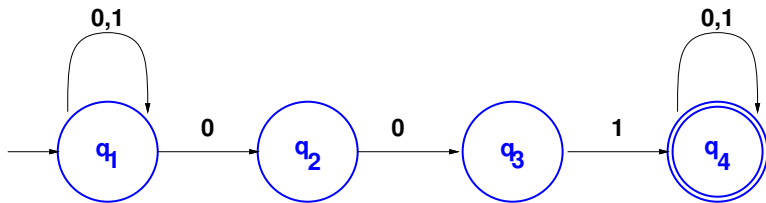
## NFA — non-deterministic Finite Automata



- ▶ May have **more than one transition** labeled with the same symbol,
- ▶ May have **no transitions** labeled with a certain symbol,
- ▶ May have transitions labeled with  $\epsilon$ , the symbol of the **empty string**. **Will deal with this latter**

Every **DFA** is also an NFA.

## Non-deterministic computation

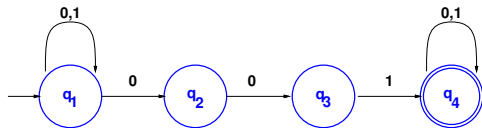


What happens when more than one transition is possible?

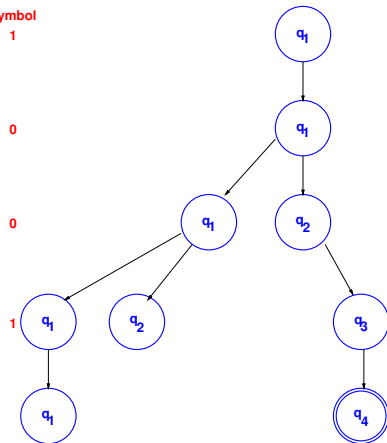
- ▶ The machine “splits” into **multiple copies**
- ▶ Each branch follows one possibility
- ▶ Together, branches follow **all** possibilities.
- ▶ If the input doesn't appear, that branch “dies”.
- ▶ Automaton accepts if **some** branch accepts.



# Computation on 1001



symbol  
1



## Why non-determinism?

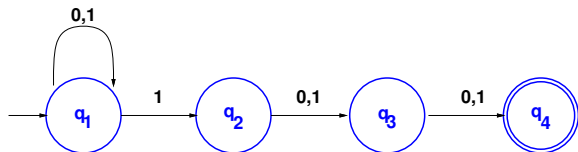
### Theorem 6 (Informal, to be proved soon)

*Deterministic and non-deterministic finite automata, **accept** exactly the **same set of languages**.*

**Q.:** So **why** do we need NFA's?

Design a finite automaton for the language  $\mathcal{L}$  — all binary strings with a **1** in their **third-to-the-last** position?

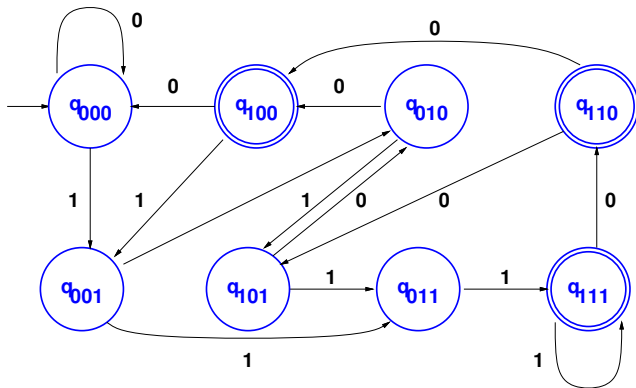
## NFA for $\mathcal{L}$



- ▶ “Guesses” which symbol is third from the last, and
- ▶ checks that indeed it is a 1.
- ▶ If guess is premature, that branch “dies”, and no harm occurs.

## DFA for $\mathcal{L}$

- ▶ Have 8 states, encoding the last three observed letters.
- ▶ A state for each string in  $\{0, 1\}^3$ .
- ▶ Add transitions on modifying the suffix, give the new letter.
- ▶ Mark as accepting, the strings  $1**$



DFA has few bugs...

## NFA – Formal Definition

Let  $\mathcal{P}(Q)$  denote the powerset of  $Q$  (i.e., all subsets of  $Q$ ).

### Definition 7 (NFA)

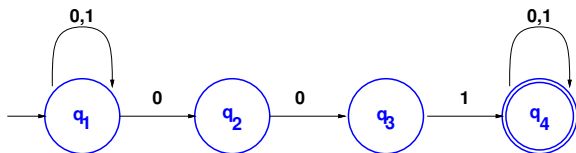
A non-deterministic finite automaton is a 5-tuple  $(Q, \Sigma, \delta, S, F)$ , where

- ▶  $Q$  is a finite set called the states
- ▶  $\Sigma$  is a finite set called the alphabet
- ▶  $\delta: Q \times \Sigma \mapsto \mathcal{P}(Q)$  is the transition function
- ▶  $S \subseteq Q$  is the set of starting states
- ▶  $F \subseteq Q$  are the set of accepting states

We sometimes consider an NFA  $(Q, \Sigma, \delta, q_0, F)$ .

This is merely a “syntactic sugar” for the NFA  $(Q, \Sigma, \delta, \{q_0\}, F)$

## Example



$$N_1 = (Q = \{q_1, q_2, q_3, q_4\}, \Sigma = \{0, 1\}, \delta, S = \{q_1\}, F = \{q_4\})$$

for  $\delta$  defined by

	0	1
$q_1$	$\{q_1, q_2\}$	$\{q_1\}$
$q_2$	$\{q_3\}$	$\emptyset$
$q_3$	$\emptyset$	$\{q_4\}$
$q_4$	$\{q_4\}$	$\{q_4\}$

Not that  $\emptyset$  is a valid output for  $\delta$

## Formal model of computation

### Definition 8

$N = (Q, \Sigma, \delta, S, F)$  **accepts**  $w \in \Sigma^*$ , if  $\hat{\delta}_N(S, w) \cap F \neq \emptyset$ .

### Definition 9 ( $\hat{\delta}$ )

For NFA  $N = (Q, \Sigma, \delta, S, F)$ , define  $\hat{\delta}_N: P(Q) \times \Sigma^* \mapsto P(Q)$  by:

$$\text{for } Q' \subseteq Q, \hat{\delta}_N(Q', w) = \begin{cases} Q', & w = \varepsilon, \\ \bigcup_{q \in \hat{\delta}_N(Q', w_1, \dots, w_{n-1})} \delta(q, w_n), & n = |w| \geq 1. \end{cases}$$

When clear from the context we will write  $\hat{\delta}$  (i.e., omitting the  $N$ ).

## An equivalent definition

### Definition 10 (Equivalent definition)

$N = (Q, \Sigma, \delta, S, F)$  accepts  $w = w_1, \dots, w_n \in \Sigma^n$ , if if  $\exists r_0, \dots, r_n \in Q$  s.t.

- ▶  $r_0 \in S$
- ▶  $r_n \in F$
- ▶  $r_{i+1} \in \delta(r_i, w_{i+1})$ , for all  $0 \leq i < n$ .



## Equivalence of NFA's and DFA's

Easy: For any DFA  $M$  there exists a NFA  $N$  such that  $\mathcal{L}(N) = \mathcal{L}(M)$ .

Other direction is also true.

### Theorem 11

*For any NFA  $N$  there exists a DFA  $M$  such that  $\mathcal{L}(N) = \mathcal{L}(M)$ .*

- ▶ Given an NFA  $N$ , we construct a DFA  $M$ , that accepts the same language.
- ▶ Make DFA emulate all possible NFA states.
- ▶ As consequence of the construction, if the NFA has  $k$  states, the DFA has  $2^k$  states (an exponential blow up).

## Equivalence of NFA's and DFA's, the DFA

Let  $N = (Q, \Sigma, \delta, S, F)$ .

**Construction 12** ( $M = (Q_M, \Sigma, \delta_M, d_0, F_M)$ )

- ▶  $Q_M = \{[R] : R \subseteq Q\}$ .
- ▶  $d_0 = [S]$
- ▶  $F_M = \{[R] \in Q_M : R \cap F \neq \emptyset\}$
- ▶ For  $[R] \in Q_M$  and  $\sigma \in \Sigma$ , let  $\delta_M([R], \sigma) = [\widehat{\delta}_N(R, \sigma)] \quad (= [\bigcup_{r \in R} \delta(r, \sigma)])$ .

To prove equivalence, we need to prove that

$$\widehat{\delta}_N(S, w) \cap F \neq \emptyset \iff \widehat{\delta}_M(d_0, w) \in F_M$$

The above is an immediate corollary of the following claim:

**Claim 13**

$[\widehat{\delta}_N(S, w)] = \widehat{\delta}_M(d_0, w)$  for every  $w \in \Sigma^*$ .

## Proving $[\widehat{\delta}_N(\mathcal{S}, w)] = \widehat{\delta}_M(d_0, w)$

The proof is by induction on the length of  $w$ .

- ▶  $|w| = 0$ , by definition.
- ▶ Assume for words of length  $(m-1)$ , and let  $x = y\sigma$ , where  $y$  is a word of length  $(m-1)$  and  $\sigma \in \Sigma$ .
- ▶ Let  $Q_y = \widehat{\delta}_N(\mathcal{S}, y)$  and  $d_y = \widehat{\delta}_M(d_0, y)$ .
- ▶ Compute

$$\widehat{\delta}_M(d_0, x) = \delta_M(d_y, \sigma) \quad (\text{By definition of } \widehat{\delta}_M)$$

$$= \delta_M([Q_y], \sigma) \quad (\text{By i.h.})$$

$$= [\widehat{\delta}_N(Q_y, \sigma)] \quad (\text{By definition of } \delta_M)$$

$$= \left[ \bigcup_{q \in Q_y} \delta(q, \sigma) \right] \quad (\text{By definition of } \widehat{\delta}_N)$$

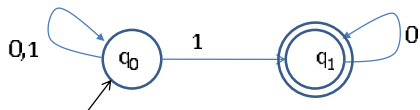
$$= \left[ \bigcup_{q \in \widehat{\delta}_N(\mathcal{S}, y)} \delta(q, \sigma) \right]$$

$$= [\widehat{\delta}_N(\mathcal{S}, x)]. \quad \square \quad (\text{By definition of } \widehat{\delta}_N)$$

- ▶ But how come we encode **infinite** parallel “threads” into a **single** thread?

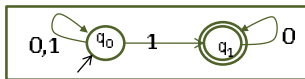
## Example: NFA $\Rightarrow$ DFA

Non-Deterministic Automata:



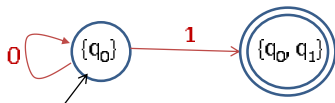
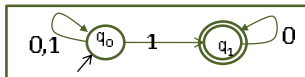
## Example: NFA $\Rightarrow$ DFA

Deterministic automata - set of states:



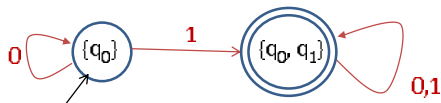
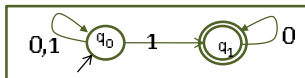
## Example: NFA $\Rightarrow$ DFA

Transitions from  $\{\{q_0\}\}$ :



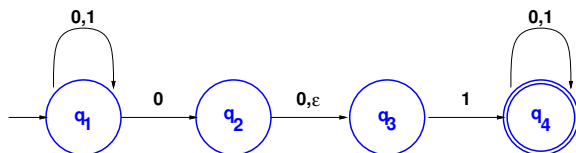
## Example: NFA $\Rightarrow$ DFA

Transitions from  $[\{q_0, q_1\}]$ :



Transitions from  $[\emptyset]$  and  $[\{q_1\}]$ ?

## NFA with $\epsilon$ -moves



What is the interpretation of  $\epsilon$  transitions ?

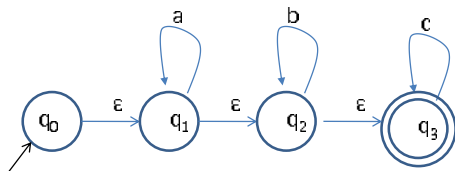
What will happen with **101** ?

How to distinguish  $\epsilon$  (that stands for the empty word) from the  $\epsilon$  above (that stands “free transition”)



## Example: NFA with $\epsilon$ -moves

$$\mathcal{L} = \{a^i b^j c^k \mid i, j, k \geq 0\}$$



## NFA — Formal definition with $\varepsilon$ -moves

Transition function  $\delta$  is going to be different.

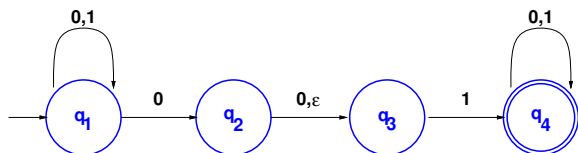
- ▶ Let  $\mathcal{P}(Q)$  denote the powerset of  $Q$ .
- ▶ Let  $\Sigma_\varepsilon$  denote the set  $\Sigma \cup \{\varepsilon\}$ .

### Definition 14 (NFA, with $\varepsilon$ -moves)

A non-deterministic finite automaton is a 5-tuple  $(Q, \Sigma, \delta, S, F)$ :

- ▶  $Q$  is a finite set called the states
- ▶  $\Sigma$  is a finite set called the alphabet
- ▶  $\delta : Q \times \Sigma_\varepsilon \mapsto \mathcal{P}(Q)$  is the transition function
- ▶  $S \subseteq Q$  is the set of starting state
- ▶  $F \subseteq Q$  is the set of accepting states

## Example



$N_1 = (Q, \Sigma, \delta, S, F)$ :

- ▶  $Q = \{q_1, q_2, q_3, q_4\}$ ,  $\Sigma = \{0, 1\}$ ,  $S = \{q_1\}$  and  $F = \{q_4\}$ .

▶  $\delta$  is

	0	1	$\epsilon$
$q_1$	$\{q_1, q_2\}$	$\{q_1\}$	$\emptyset$
$q_2$	$\{q_3\}$	$\emptyset$	$\{q_3\}$
$q_3$	$\emptyset$	$\{q_4\}$	$\emptyset$
$q_4$	$\{q_4\}$	$\{q_4\}$	$\emptyset$

## Formal model of computation, with $\varepsilon$ -moves

### Definition 15

$N = (Q, \Sigma, \delta, S, F)$  **accepts**  $w \in \Sigma^*$ , if  $\widehat{\delta}_N(S, w) \cap F \neq \emptyset$ .

### Definition 16

For NFA  $N = (Q, \Sigma, \delta, S, F)$ , let

$E(q) = \{q' \in Q : q' \text{ can be reached from } q \text{ by } 0 \text{ or more } \varepsilon \text{ transitions}\}$   
(i.e.,  $\{q' : \exists q_1, \dots, q_k \in Q \text{ s.t. } q_1 = q \wedge q_k = q' \wedge \forall i \in [k-1] \ q_{i+1} \in \delta(q_i, \varepsilon)\}$ )

$E(Q') = \bigcup_{q \in Q'} E(q)$ .

**Q:** is it always the case that  $q \in E(q)$ ? Yes

### Definition 17 ( $\widehat{\delta}$ )

For NFA  $N = (Q, \Sigma, \delta, S, F)$ , define  $\widehat{\delta}_N: P(Q) \times \Sigma^* \mapsto P(Q)$  by:

for  $Q' \subseteq Q$ ,  $\widehat{\delta}_N(Q', w) = \begin{cases} E(Q'), & w = \varepsilon, \\ E\left(\bigcup_{r \in \widehat{\delta}(Q', w_1, \dots, w_{n-1})} \delta(r, w_n)\right), & n = |w| \geq 1. \end{cases}$

When does  $N$  accept the empty string?

## An equivalent definition

For  $a \in (\Sigma_\varepsilon)^*$ , let  $d(a) \in \Sigma^*$  be  $a$  without the  $\varepsilon$  symbols.

Example:  $d(\varepsilon 01 \varepsilon \varepsilon 3 \varepsilon) = 013$

### Definition 18 (Equivalent definition)

$N = (Q, \Sigma, \delta, S, F)$  accepts  $w \in \Sigma^*$ , if exist  $a = (a_1 a_2 \dots a_k) \in (\Sigma_\varepsilon)^k$  and  $r_0, \dots, r_k \in Q$  s.t.

- ▶  $w = d(a)$ .
- ▶  $r_0 \in S$
- ▶  $r_k \in F$
- ▶  $r_{i+1} \in \delta(r_i, a_{i+1})$ , for all  $0 \leq i < k$ .

## Removing $\varepsilon$ -transitions

Given NFA  $N = (Q, \Sigma, \delta, S, F)$  with  $\varepsilon$ -transitions, we create an **equivalent** NFA  $N' = (Q, \Sigma, \delta', S', F)$  with **no**  $\varepsilon$ -transitions.

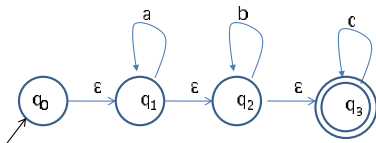
- ▶  $S' = E(S)$
- ▶  $\delta'(q, a) = E(\delta(q, a))$

It is not hard to prove that  $\widehat{\delta}_N(S, w) = \widehat{\delta}_{N'}(S', w)$  for any  $w \in \Sigma^*$ .

Thus,  $\mathcal{L}(N) = \mathcal{L}(N')$ .

## Example: Removing $\epsilon$ -transitions

Non-Deterministic Automata with  $\epsilon$ -transitions



The non-Deterministic automata without  $\epsilon$ -transitions

►  $S' = \{q_0, q_1, q_2, q_3\}$

►  $\delta'$  is

	a	b	c
$q_0$	$\emptyset$	$\emptyset$	$\emptyset$
$q_1$	$\{q_1, q_2, q_3\}$	$\emptyset$	$\emptyset$
$q_2$	$\emptyset$	$\{q_2, q_3\}$	$\emptyset$
$q_3$	$\emptyset$	$\emptyset$	$\{q_3\}$

# Part II

## Closure of Regular Languages, Revisited



## Regular languages, revisited

By definition, a language is regular if it is accepted by some **DFA**.

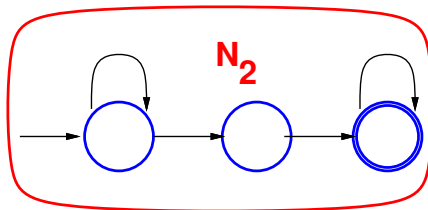
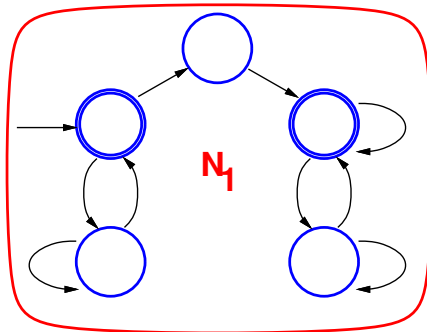
### Corollary 19

*A language is regular if and only if it is accepted by some **NFA**.*

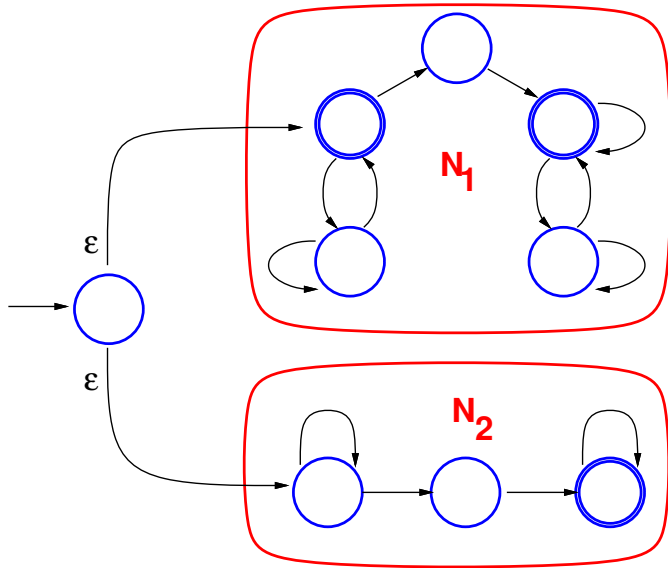
This is an alternative way of characterizing regular languages.

We will now use the equivalence to show that regular languages are **closed** under the regular operations (union, concatenation, star).

## Closure under **union** (alternative proof)



## Closure under union, cont.



## Closure under union cont..

- ▶ NFA  $N_1 = (Q_1, \Sigma, \delta_1, S_1, F_1)$  accept  $\mathcal{L}_1$ , and
- ▶ NFA  $N_2 = (Q_2, \Sigma, \delta_2, S_2, F_2)$  accept  $\mathcal{L}_2$ .

wlg.  $Q_1 \cap Q_2 = \emptyset$ .(?)

Define NFA  $N = (Q = \{q_0\} \cup Q_1 \cup Q_2, \Sigma, \delta, S = \{q_0\}, F = F_1 \cup F_2)$ ,

$$\text{for } \delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ S_1 \cup S_2 & q = q_0 \text{ and } a = \varepsilon \end{cases}$$

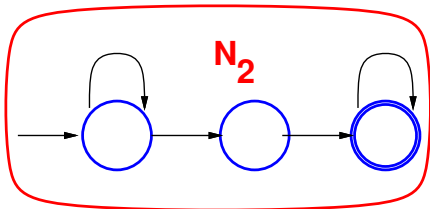
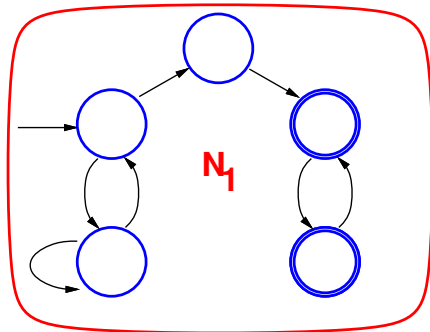
Alternatively, let  $S = S_1 \cup S_2$  and omit the last line of  $\delta$ .

### Claim 20

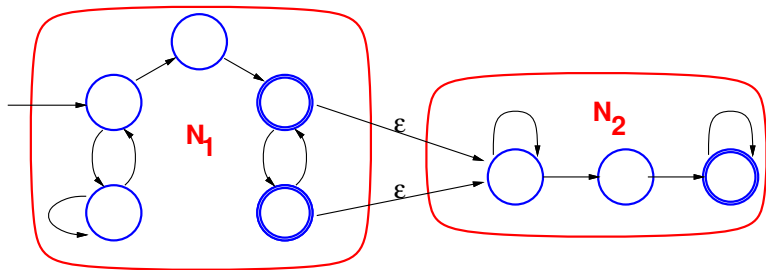
$$\mathcal{L}(N) = \mathcal{L}(N_1) \cup \mathcal{L}(N_2).$$

Proof: Easily follow by the the alternative definition of acceptance by NFA, DIY...

## Closure under concatenation



## Closure under concatenation, cont.



**Remark:** Final states are exactly those of  $N_2$ .

## Closure under concatenation, cont..

- ▶ NFA  $N_1 = (Q_1, \Sigma, \delta_1, S_1, F_1)$  accept  $\mathcal{L}_1$
- ▶ NFA  $N_2 = (Q_2, \Sigma, \delta_2, S_2, F_2)$  accept  $\mathcal{L}_2$

Define NFA  $N = (Q_1 \cup Q_2, \Sigma, \delta, S_1, F_2)$ :

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \wedge a \neq \varepsilon \\ \delta_1(q, a) & (q \in Q_1 \setminus F_1) \wedge a = \varepsilon \\ \delta_1(q, a) \cup S_2 & q \in F_1 \wedge a = \varepsilon \\ \delta_2(q, a) & q \in Q_2 \end{cases}$$

### Claim 21

$$\mathcal{L}(N) = \mathcal{L}_1 \parallel \mathcal{L}_2.$$

Proof: Need to prove  $w \in \mathcal{L}(N) \iff w \in \mathcal{L}_1 \parallel \mathcal{L}_2$ .

## Proving $w \in \mathcal{L}(N) \iff \mathcal{L}_1 \parallel \mathcal{L}_2$

Let  $w \in \mathcal{L}_1 \parallel \mathcal{L}_2$ :

$\implies \exists w^1 \in \mathcal{L}_1$  and  $w^2 \in \mathcal{L}_2$ , s.t.  $w = w^1 w^2$

$\implies \exists r_0^1, \dots, r_{|w^1|}^1$  and  $r_0^2, \dots, r_{|w^2|}^2$ , such that for both  $j \in \{1, 2\}$ :

(1)  $r_0^j \in S_j$  (2)  $r_{|w^j|}^j \in F_j$  (3)  $\forall 0 \leq i < |w^j|: r_{i+1}^j \in \delta_j(r_i^j, w_{i+1}^j)$ .

$\implies$  (details...)  $r_0^1, \dots, r_{|w^1|}^1, r_0^2, \dots, r_{|w^2|}^2$  proves that  $r_{|w^2|}^2 \in \widehat{\delta}(S_1, w)$

$\implies w \in \mathcal{L}(N)$

Let  $w \in \mathcal{L}(N)$ :

$\implies \exists w' = a_1 a_2 \dots a_k \in (\Sigma_\varepsilon)^k$  and  $r_0 \dots r_k$  such that:

(1)  $w = d(w')$  (2)  $r_0 \in S_1$  (3)  $r_k \in F_2$  (4)  $\forall 0 \leq i < k: r_{i+1} \in \delta(r_i, a_{i+1})$ .

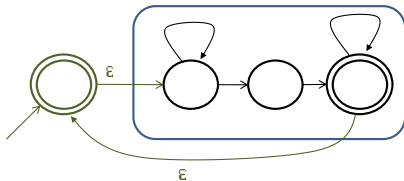
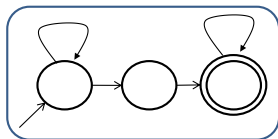
► Let  $j$  be the **last** index such that  $r_j \in Q_1$

► By construction (details...) (1)  $a_{j+1} = \varepsilon$  (2)  $r_0 \dots r_j$  proves that  $d(a_1, \dots, a_j) \in \mathcal{L}_1$  (3)  $r_{j+1} \dots r_k$  proves that  $d(a_{j+2}, \dots, a_k) \in \mathcal{L}_2$

$\implies w = d(w' = a_1, \dots, a_j, \varepsilon, a_{j+2}, \dots, a_k) = d(a_1, \dots, a_j) d(a_{j+2}, \dots, a_k) \in \mathcal{L}_1 \parallel \mathcal{L}_2 \quad \square$



## Closure under star



## Closure under **star**, cont.

Let  $N = (Q, \Sigma, \delta, S, F)$  accepting  $\mathcal{L}$ , assuming wlg. that  $q_0 \notin Q$ .

Define  $N' = (Q' = Q \cup \{q_0\}, \Sigma, \delta', S' = \{q_0\}, F' = \{q_0\})$ :

$$\delta'(q, a) = \begin{cases} \delta(q, a) & q \in Q \wedge a \neq \varepsilon \\ \delta(q, \varepsilon) & q \notin F \wedge a = \varepsilon \\ \delta(q, \varepsilon) \cup \{q_0\} & q \in F \wedge a = \varepsilon \\ S & q = q_0 \wedge a = \varepsilon \end{cases}$$

### Claim 22

$$\mathcal{L}(N') = \mathcal{L}(N)^*.$$

Proof: DIY

# Summary

- ▶ **Regular languages** are closed under
  - ▶ union
  - ▶ concatenation
  - ▶ star
- ▶ **Non-deterministic** finite automata
  - ▶ are equivalent to **deterministic** finite automata
  - ▶ but much easier to use in some proofs and constructions.

# Part III

## Regular Expressions

## Regular expressions

Notation for building up languages by describing them as expressions, e.g.,  $(0 \cup 1)0^*$ .

- ▶  $0$  and  $1$  are shorthand for the set (languages)  $\{0\}$  and  $\{1\}$
- ▶ so  $0 \cup 1 = \{0, 1\}$ .
- ▶  $0^*$  is shorthand for  $\{0\}^*$ .
- ▶ Concatenation, is implicit. So  $0^*10^*$  stands for  $\{w \in \{0, 1\}^* : w \text{ has exactly a single } 1\}$ .
- ▶ Just like in arithmetic, operations have precedence:
  - ▶ star first
  - ▶ concatenation next
  - ▶ union last
  - ▶ parentheses used to change default order i.e.,  $ab^* \neq (ab)^*$

**Q.:** What does  $(0 \cup 1)0^*$  stand for?

**Remark:** Regular expressions are often used in text editors or shell scripts.

## Regular expressions – formal definition

### Definition 23

A string  $R$  is a **regular expression** over alphabet  $\Sigma$ , if  $R$  is of form

- ▶  $a$  for some  $a \in \Sigma$
- ▶  $\varepsilon$
- ▶  $\emptyset$
- ▶  $(R_1 \cup R_2)$  for regular expressions  $R_1$  and  $R_2$
- ▶  $(R_1 \| R_2)$  for regular expressions  $R_1$  and  $R_2$
- ▶  $(R_1^*)$  for regular expression  $R_1$

$R(\Sigma)$  denotes all (finite) regular expression over  $\Sigma$ .

Parenthesis and  $\|$  are omitted when their role is clear from the context.

## Formal definition, cont.

### Definition 24

The language  $\mathcal{L}(R)$  of regular expression  $R$ , is defined by

$R$	$\mathcal{L}(R)$
$a$	$\{a\}$
$\varepsilon$	$\{\varepsilon\}$
$\emptyset$	$\emptyset$
$(R_1 \cup R_2)$	$\mathcal{L}(R_1) \cup \mathcal{L}(R_2)$
$(R_1 R_2)$	$\mathcal{L}(R_1) \parallel \mathcal{L}(R_2)$
$(R_1^*)$	$\mathcal{L}(R_1)^*$

Isn't this definition circular?

## Examples of regular expressions

For  $\Sigma = \{0, 1\}$ , write regular expression for the following languages:

- ▶ The third letter from the end is 1

$$(0 \cup 1)^* 1 (0 \cup 1)^2$$

- ▶ The number of 1's is even

$$(0^* \cup 10^* 1)^*$$

- ▶ The number of 1's is odd

$$(0^* \cup 10^* 1)^* 10^*$$



# Part IV

## Regular Expressions and Regular Languages

## A remarkable fact

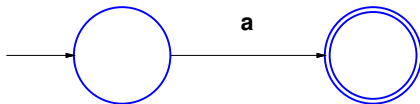
### Theorem 25

*A language is described by a regular expression iff it is regular.*

$\Leftarrow$ : Given a **regular language**, construct a **regular expression** describing it.

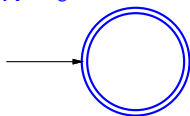
$\Rightarrow$ : Given a **regular expression**, construct an **NFA** accepting its language.

## Given RE $R$ , build NFA Accepting it ( $\implies$ )

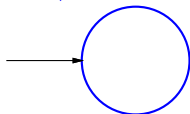


1.  $R = a$ , for some  $a \in \Sigma$

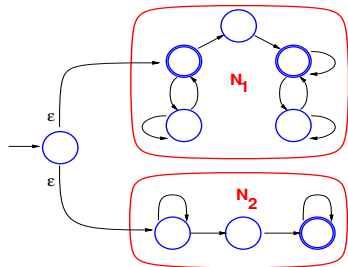
2.  $R = \epsilon$



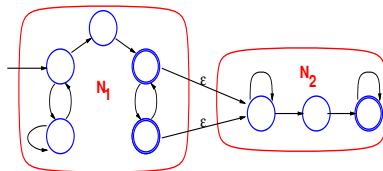
3.  $R = \emptyset$



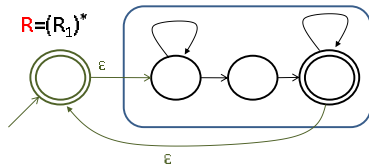
# Given $R$ , Build NFA Accepting It ( $\implies$ ), cont.



$$R = (R_1 \cup R_2)$$

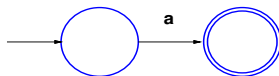


$$R = (R_1 || R_2)$$

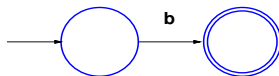


## Examples

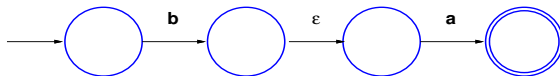
**a**



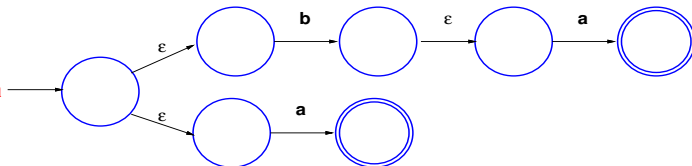
**b**



**ba**



**ba  $\cup$  a**



Formal proof by induction on the length of the regular expression

## Regular expression from a DFA ( $\Leftarrow$ )

Easy for DFAs that accept finite languages, but more complicated for general DFA's.

### NFA:

- ▶ Each transition is labeled with a symbol or  $\epsilon$ .
- ▶ Reads **zero or one** symbols.
- ▶ Takes matching transition, if any.

### Generalized non-deterministic finite automata (GNFA):

- ▶ Each transition is labeled with a **regular expression**.
- ▶ Reads **zero or more** symbols.
- ▶ Takes matching **regular expression**, if any.

Example (board).

GNFAs are natural generalization of NFAs.

## GNFA – Formal Definition

Let  $\mathcal{R}(\Sigma)$  be the set of regular expressions over  $\Sigma$ .

### Definition 26

A **generalized** deterministic finite automaton (GNFA) is  $(Q, \Sigma, \delta, q_s, q_a)$

- ▶  $Q$  is a finite set of **states**
- ▶  $\Sigma$  is the **alphabet**
- ▶  $\delta : (Q \setminus \{q_a\}) \times (Q \setminus \{q_s\}) \mapsto \mathcal{R}(\Sigma)$  is the **transition function**.
- ▶  $q_s \in Q$  is the **start state**
- ▶  $q_a \in Q$  is the unique **accept state**

It is a special type of GNFA, but still it is easy to transform any DFA/NFA into this form.

### Definition 27

A GNFA  $G = (Q, \Sigma, \delta, q_s, q_a)$  accepts a string  $w \in \Sigma^*$ , if there exists

- ▶ parsing  $w = a_1 a_2 \cdots a_k \in (\Sigma^*)^k$ , and
- ▶  $r_0, \dots, r_k \in Q$ ,

such that

- ▶  $r_0 = q_s$
- ▶  $r_k = q_a$
- ▶  $a_i \in \mathcal{L}(\delta(r_{i-1}, r_i))$ , for every  $0 < i \leq k$ .



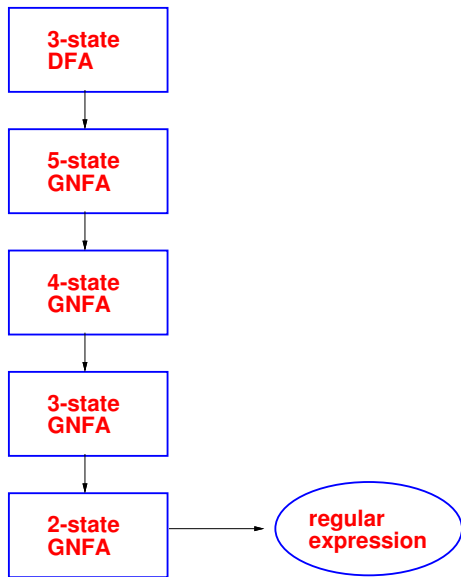
## The Transformation: DFA $\rightarrow$ Regular Expression

Strategy – sequence of **equivalent** transformations

- ▶ Given a  $k$ -state DFA
- ▶ Transform into  $(k + 2)$ -state GNFA (how?)
- ▶ While GNFA has **more than 2 states**, transform it into equivalent GNFA with **one fewer** state
- ▶ Eventually reach 2-state GNFA (states are just **start** and **accept**).
- ▶ Label of **single** transition is the desired regular expression.

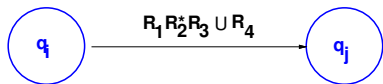
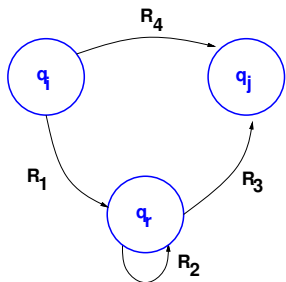


## Converting strategy ( $\Leftarrow$ )

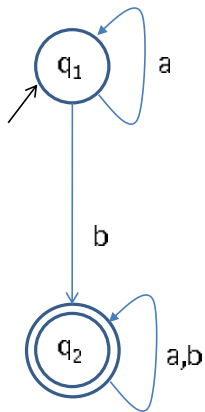


## Removing a state

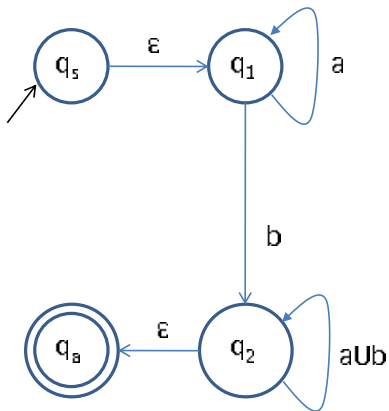
We **remove** one state  $q_r$ , and then **repair** the machine by **altering** regular expression of other transitions.



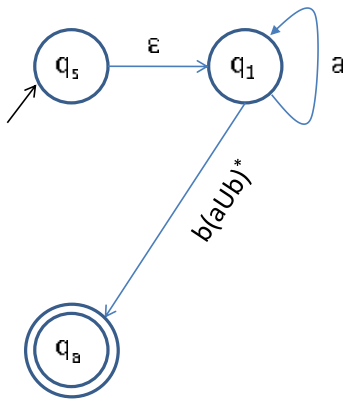
## Conversion - Example



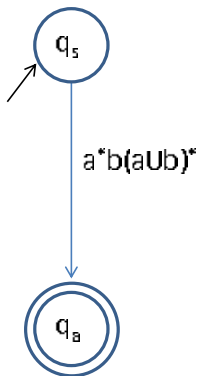
## Conversion - Example



## Conversion - Example



## Conversion - Example



## The StateReduce and Convert algorithms

### Algorithm 28 (StateReduce)

Input: a  $(k > 2)$ -state GNFA  $G = (Q, \Sigma, \delta, q_s, q_a)$ .

- ▶ Select any state  $q_r \in Q \setminus \{q_s, q_a\}$ .
- ▶ Let  $Q' = Q \setminus \{q_r\}$ .
- ▶ For any  $q_i \in Q' \setminus \{q_a\}$  and  $q_j \in Q' \setminus \{q_s\}$ , let
  - ▶  $R_1 = \delta(q_i, q_r)$ ,  $R_2 = \delta(q_r, q_r)$ ,  $R_3 = \delta(q_r, q_j)$  and  $R_4 = \delta(q_i, q_j)$ .
- ▶ Define  $\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4)$ .
- ▶ Return the resulting  $(k - 1)$ -state GNFA  $G' = (Q', \Sigma, \delta', q_s, q_a)$ .

### Algorithm 29 (Convert)

Input: a  $(k \geq 2)$ -state GNFA  $G$ .

- ▶ If  $k = 2$ , return the regular expression labeling the only arrow of  $G$ .
- ▶ Otherwise, return **Convert**(StateReduce( $G$ )).



## Correctness proof

### Claim 30

$G$  and  $\text{Convert}(G)$  accept the same language.

Proof: By induction on  $k$  – the number of states of  $G$ .

**Basis.**  $k = 2$ : Immediate by the definition of GFNA.

**Induction step:** Assume claim for  $(k - 1)$ -state GNFA, where  $k > 2$ , prove for  $k$ -state GNFA.

Let  $G' = \text{StateReduce}(G)$  (note that  $G'$  has  $k - 1$  states), and let  $q_r$  be the removed state.

We prove (in a very high level) that  $\mathcal{L}(G) = \mathcal{L}(G')$  (i.e.,  $G$  and  $G'$  accept the same language). We show

- ▶  $w \in \mathcal{L}(G) \implies w \in \mathcal{L}(G')$
- ▶  $w \in \mathcal{L}(G') \implies w \in \mathcal{L}(G)$

$$w \in \mathcal{L}(G) \implies w \in \mathcal{L}(G')$$

Let  $w \in \mathcal{L}(G)$  and let  $p = q_s, \dots, q_a$  be (a possible) “path of states” traversed by  $G$  on  $w$ .

- ▶ If  $q_r \notin p$ , then  $G'$  accepts  $w$  (the new regular expression on each edge of  $G'$  contains the old regular expression in the “union part”.)
- ▶ If  $p = q_s, \dots, q_i, q_r, q_j, \dots, q_a$ , the regular expression  $(R_{i,r})(R_{r,r})^*(R_{r,j})$  linking  $q_i$  and  $q_j$  in  $G'$ , causes  $G'$  to accept  $w$ .

Hence,  $w \in \mathcal{L}(G')$ .

$$w \in \mathcal{L}(G') \implies w \in \mathcal{L}(G)$$

- ▶ Let  $q_s = q_{j_0}, q_{j_1}, \dots, q_{j_n} = q_a$  be (a possible) “path of states” traversed by  $G'$  on  $w$ .

$\exists$  parsing  $w = w_1, \dots, w_n$  s.t.  $w_i \in \mathcal{L}(\delta_{G'}(q_{j_{i-1}}, q_{j_i}))$ .

- ▶ For every  $i \in [n]$ :

$$\delta_{G'}(q_{j_{i-1}}, q_{j_i}) = (\delta_G(q_{j_{i-1}}, q_r) \delta_G(q_r, q_r)^* \delta_G(q_r, q_{j_i})) \cup (\delta_G(q_{j_{i-1}}, q_{j_i})).$$

- ▶ Hence for all  $i \in [n]$ , the word  $w_i$  corresponds to possible traverse from  $q_{j_{i-1}}$  to  $q_{j_i}$  in  $G$ .
- ▶ Hence,  $w \in \mathcal{L}(G') \implies w \in \mathcal{L}(G)$ .

## Summing it up

- ▶ We proved  $\mathcal{L}(G) = \mathcal{L}(G')$ .
- ▶ Hence,  $G$  and (the regular expression)  $\text{Convert}(G)$  accept the same language.
- ▶ Thus, we proved: Every regular language can be described by a **regular expression**.

# Summary

- ▶ Non-Deterministic Automata (with  $\epsilon$ -moves)
  - ▶ Equivalence to DFA
- ▶ Closure properties
  - ▶ union
  - ▶ concatenation
  - ▶ star
- ▶ Regular expressions.
  - ▶ Equivalence to DFA