Damascus University

Automata theory and formal languages

Regular Expressions

Dr. Mohammad Ahmad

Regular Expressions vs. Finite Automata

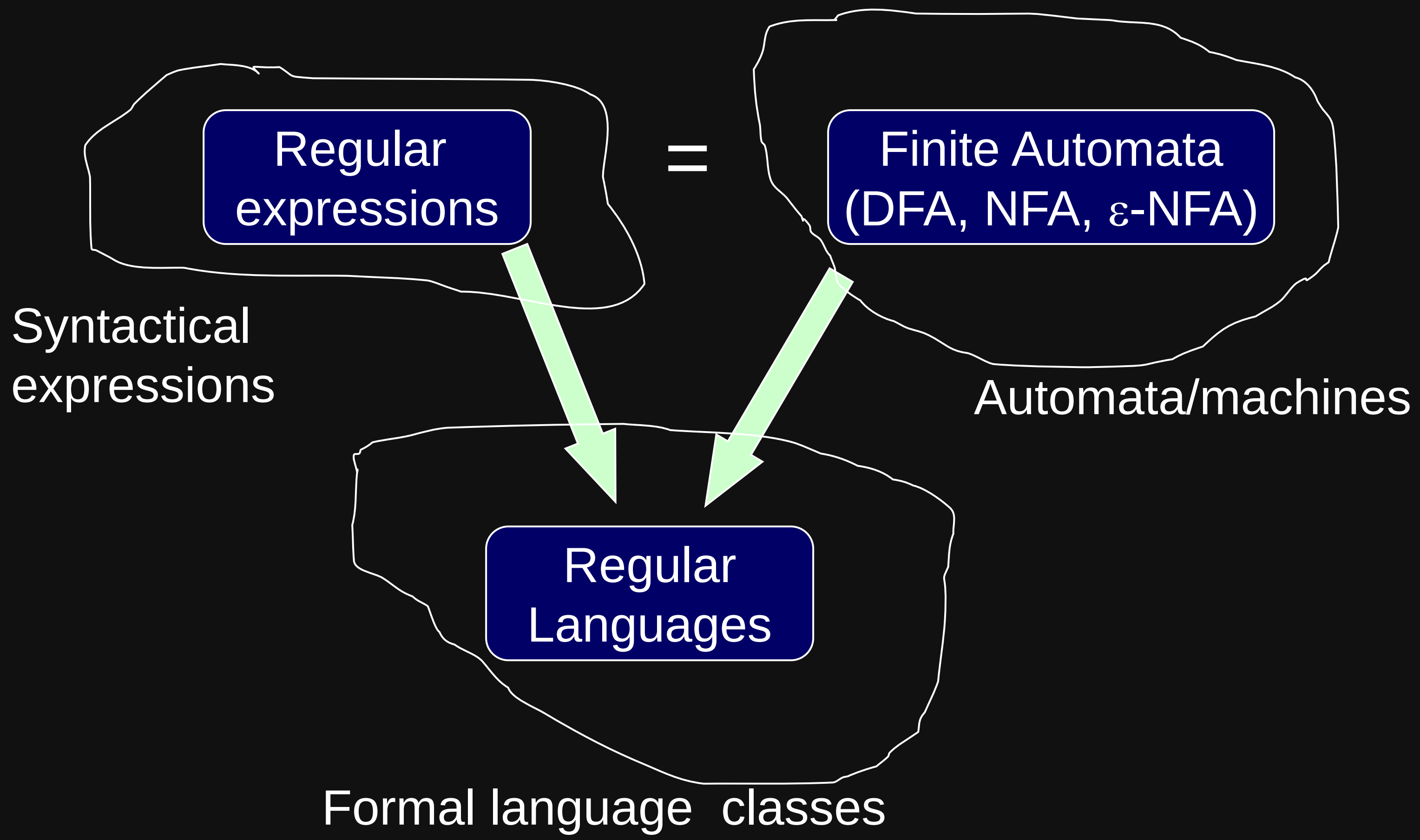
- Offers a declarative way to express the pattern of any string we want to accept
  - E.g., **01*+ 10***

- Automata = more machine-like

  < input: string , output: [accept/reject] >
- Regular expressions = more program syntax-like

- Unix environments heavily use regular expressions
  - E.g., bash shell, grep, vi & other editors, sed
- Perl scripting – good for string processing
- Lexical analyzers such as Lex or Flex

Regular Expressions



String concatenation

$s = 011$ $\quad t = 101$

$st = 011101$

$ts = 101011$

$ss = 011011$

$sst = 011011101$

$s = \mathrm{a}_1 \ldots \mathrm{a}_n \quad t = \mathrm{b}_1 \ldots \mathrm{b}_m \Rightarrow st = \mathrm{a}_1 \ldots \mathrm{a}_n \mathrm{b}_1 \ldots \mathrm{b}_m$

Language Operators

- Union of two languages:
  - **L U M** = all strings that are either in L or M
  - Note: A union of two languages produces a third language

- Concatenation of two languages:
  - **L . M** = all strings that are of the form *xy* s.t., $x \in L$ and $y \in M$
  - The *dot* operator is usually omitted
    - i.e., **LM** is same as L.M

Kleen

"i" here refers to how many strings to concatenate from the parent language L to produce strings in the language L^i

- Kleene Closure of a given language L:
  - $L^0 = \{\varepsilon\}$
  - $L^1 = \{w \mid \text{for some } w \in L\}$
  - $L^2 = \{ w_1w_2 \mid w_1 \in L, w_2 \in L \text{ (duplicates allowed)}\}$
  - $L^i = \{ w_1w_2\ldots w_i \mid \text{all } w\text{'s chosen are} \in L \text{ (duplicates allowed)}\}$
  - (Note: the choice of each w_i is independent)
  - $\mathbf{L^*} = \bigcup_{i\geq 0} L^i$ (arbitrary number of concatenations)

Example:

- Let L = { 1, 00}
  - $L^0 = \{\varepsilon\}$
  - $L^1 = \{1,00\}$
  - $L^2 = \{11,100,001,0000\}$
  - $L^3 = \{111,1100,1001,10000,000000,00001,00100,0011\}$
  - $\mathbf{L^*} = L^0 \cup L^1 \cup L^2 \cup \ldots$

Kleene Closure (special notes)

- L^* is an infinite set iff $|L| \geq 1$ and $L \neq \{\varepsilon\}$

Why?

- If $L=\{\varepsilon\}$, then $L^* = \{\varepsilon\}$

Why?

- If $L = \Phi$, then $L^* = \{\varepsilon\}$

Why?

Σ^* denotes the set of all words over an alphabet Σ

- Therefore, an abbreviated way of saying there is an arbitrary language L over an alphabet Σ is:
  - $L \subseteq \Sigma^*$

Operations on languages

- The concatenation of languages L_1 and L_2 is

$$L_1L_2 = \{st\colon s \in L_1, t \in L_2\}$$

- The n-th power of L^n is

$$L^n = \{s_1s_2\ldots s_n\colon s_1, s_2, \ldots, s_n \in L\}$$

- The union of L_1 and L_2 is

$$L_1 \cup L_2 = \{s\colon s \in L_1 \text{ or } s \in L_2\}$$

Example

$$L_1 = \{0, 01\}$$

$$L_2 = \{\varepsilon, 1, 11, 111, \ldots\}$$

any number of 1s

$$\begin{aligned} L_1L_2 &= \{0, 01, 011, 0111, \ldots\} \cup \{01, 011, 0111, \ldots\} \\ &= \{0, 01, 011, 0111, \ldots\} \end{aligned}$$

0 followed by any number of 1s

$$L_1{}^2 = \{00, 001, 010, 0101\}$$

$$\begin{aligned} L_2{}^2 &= L_2 \\ L_2{}^n &= L_2 \quad (n \geq 1) \end{aligned}$$

$$L_1 \cup L_2 = \{0, 01, \varepsilon, 1, 11, 111, \ldots\}$$

Operations on languages

- The star of L are all strings made up of zero or more chunks from L:

$$L^* = L^0 \cup L^1 \cup L^2 \cup \ldots$$

  – This is always infinite, and always contains ε

- Example: $L_1 = \{01, 0\}, L_2 = \{\varepsilon, 1, 11, 111, \ldots\}$. What is $L_1{}^*$ and $L_2{}^*$?

Example

$L_1 = \{0, 01\}$

$L_2 = \{\varepsilon, 1, 11, 111, \ldots\}$

any number of 1s

$L_1^2 = \{00, 001, 010, 0101\}$

L_1^*: $0|01|0|0|0|01$ is in L_1^*

$00\underline{11}0001$ is not in L_1^*

$\underline{1}0010001$ is not in L_1^*

L_1^* are all strings that start with 0 and do not contain consecutive 1s

$L_2^2 = L_2$

$L_2^n = L_2 \quad (n \geq 1)$

$$\begin{aligned} L_2^* &= L_2^0 \cup L_2^1 \cup L_2^2 \cup \ldots \\ &= \{\varepsilon\} \cup L_2 \cup L_2 \cup \ldots \\ &= L_2 \end{aligned}$$

$L_2^* = L_2$

Constructing languages with operations

- Let's say $\Sigma = \{0, 1\}$
- We can construct languages by starting with simple ones, like $\{0\}$, $\{1\}$ and combining them

$\{0\}(\{0\}\cup\{1\})^*$ ⇒ $0(0+1)^*$

all strings that start with 0

$(\{0\}\{1\}^*)\cup(\{1\}\{0\}^*)$ ⇒ 01^*+10^*

0 followed by any number of 1s, or
1 followed by any number of 0s

Regular expressions

- A regular expression over Σ is an expression formed using the following rules:
  - The symbol $\emptyset$ is a regular expression
  - The symbol ε is a regular expression
  - For every $a \in \Sigma$, the symbol a is a regular expression
  - If R and S are regular expressions, so are $R+S$, RS and R^*.

A language is regular if it is represented by a regular expression

Examples

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

$$01^* \;=\; 0(1^*) \;=\; \{0, 01, 011, 0111, \ldots\}$$

0 followed by any number of 1s

$$(01^*)(01) \;=\; \{001, 0101, 01101, 011101, \ldots\}$$

0 followed by any number of 1s and then 01

Examples

$0+1 = \{0, 1\}$ strings of length 1

$(0+1)^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots\}$ any string

$(0+1)^*010$ any string that ends in 010

$(0+1)^*01(0+1)^*$ any string that contatins the pattern 01

Examples

$$((0+1)(0+1))^*+((0+1)(0+1)(0+1))^*$$

all strings whose length is even or a mutliple of 3
= strings of length $0, 2, 3, 4, 6, 8, 9, 10, 12, \ldots$

$((0+1)(0+1))^*$ strings of even length

$(0+1)(0+1)$ strings of length 2

$((0+1)(0+1)(0+1))^*$ strings of length a multiple of 3

$(0+1)(0+1)(0+1)$ strings of length 3

Examples

$((0+1)(0+1)+(0+1)(0+1)(0+1))^*$

strings that can be broken in blocks, where each block has length 2 or 3

$(0+1)(0+1)+(0+1)(0+1)(0+1)$ strings of length 2 or 3

$(0+1)(0+1)$ strings of length 2

$(0+1)(0+1)(0+1)$ strings of length 3

Examples

$((0+1)(0+1)+(0+1)(0+1)(0+1))^*$

strings that can be broken in blocks,
where each block has length 2 or 3

ε ✓ 1 ✗ 10 ✓ 011 ✓ $\underbrace{00}\underbrace{110}$ ✓ $\underbrace{01}\underbrace{10}\underbrace{10}\underbrace{110}$ ✓

this includes all strings except those of length 1

$((0+1)(0+1)+(0+1)(0+1)(0+1))^* =$ all strings except 0 and 1

Examples

$$\underbrace{(1+01+001)^*}\underbrace{(\varepsilon+0+00)}$$

ends in at most two 0s

there can be at most two 0s between consecutive 1s

---

there are never three consecutive 0s

Guess: $(1+01+001)^*(\varepsilon+0+00) = \{x\colon x \text{ does not contain } 000\}$

ε $\quad 00 \quad 01|1|001|01|1|0 \quad 001|001|0$

Examples

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

- Write a regular expression for all strings with two consecutive 0s.

(anything) 00 (anything else)

$\boxed{(0+1)^*00(0+1)^*}$

Examples

- Write a regular expression for all strings that do not contain two consecutive 0s.

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

$$\underbrace{01|1|01|01|1|01|01|}_{\text{blocks ending in } 1}\underbrace{0}_{\text{last block}}$$

... at most one 0 in every block ending in 1 $(1 + 01)$

... and at most one 0 in the last block $(\varepsilon + 0)$

$$\boxed{(1 + 01)^*(\varepsilon + 0)}$$

Examples

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

- Write a regular expression for all strings with an even number of 0s.

even number of zeros $=$ (two zeros)*

two zeros $= 1^*01^*01^*$

$\boxed{(1^*01^*01^*)^*}$

Building Regular Expressions

- Let E be a regular expression and the language represented by E is L(E)
- Then:
  - (E) = E
  - L(E + F) = L(E) U L(F)
  - L(E F) = L(E) L(F)
  - L(E*) = (L(E))*

Example: how to use these regular expression properties and language operators?

- *L = { w | w is a binary string which does not contain two consecutive 0s or two consecutive 1s anywhere)*
  - E.g., w = 01010101 is in L, while w = 10010 is not in L
- Goal: Build a regular expression for L
- Four cases for w:
  - Case A: w starts with 0 and |w| is even
  - Case B: w starts with 1 and |w| is even
  - Case C: w starts with 0 and |w| is odd
  - Case D: w starts with 1 and |w| is odd
- Regular expression for the four cases:
  - Case A: $(01)^*$
  - Case B: $(10)^*$
  - Case C: $0(10)^*$
  - Case D: $1(01)^*$
- Since L is the union of all 4 cases:
  - Reg Exp for L = $(01)^* + (10)^* + 0(10)^* + 1(01)^*$
- If we introduce ε then the regular expression can be simplified to:
  - Reg Exp for L = $(\varepsilon + 1)(01)^*(\varepsilon + 0)$

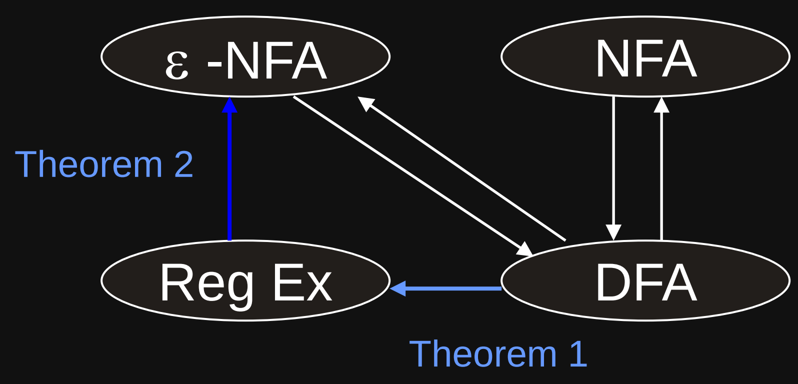
Precedence of Operators

- Highest to lowest
  - * operator (star)
  - . (concatenation)
  - \+ operator
- Example:
  - $01^* + 1 = (0 . ((1)^*)) + 1$

Finite Automata (FA) & Regular Expressions (Reg Ex)

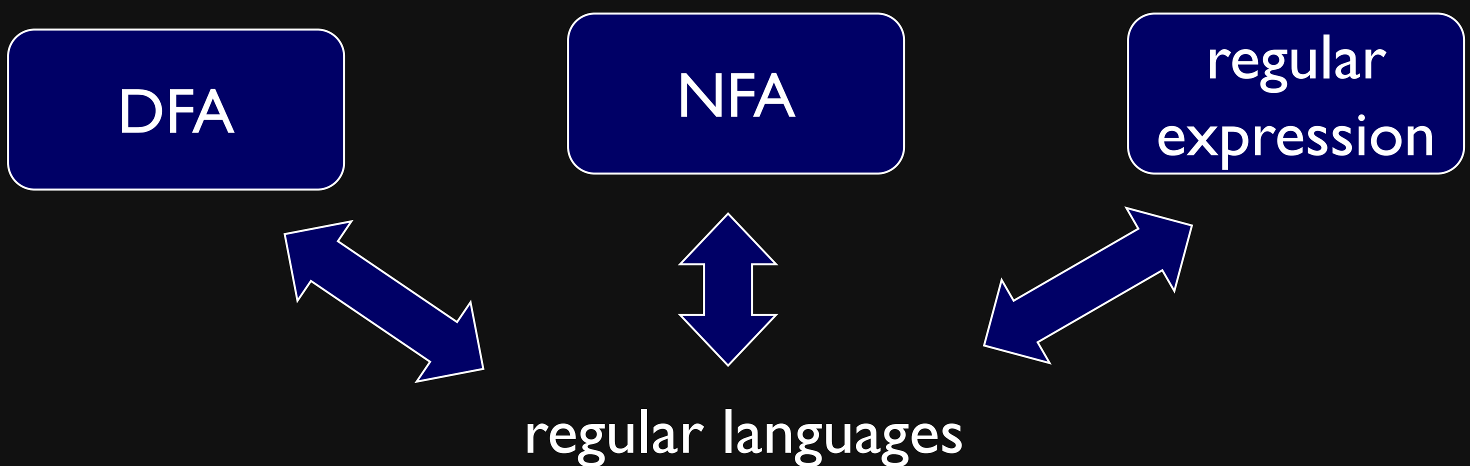
- To show that they are interchangeable, consider the following theorems:
  - *Theorem 1: For every DFA A there exists a regular expression R such that L(R)=L(A)*
  - *Theorem 2: For every regular expression R there exists an ε-NFA E such that L(E)=L(R)*

Kleene Theorem

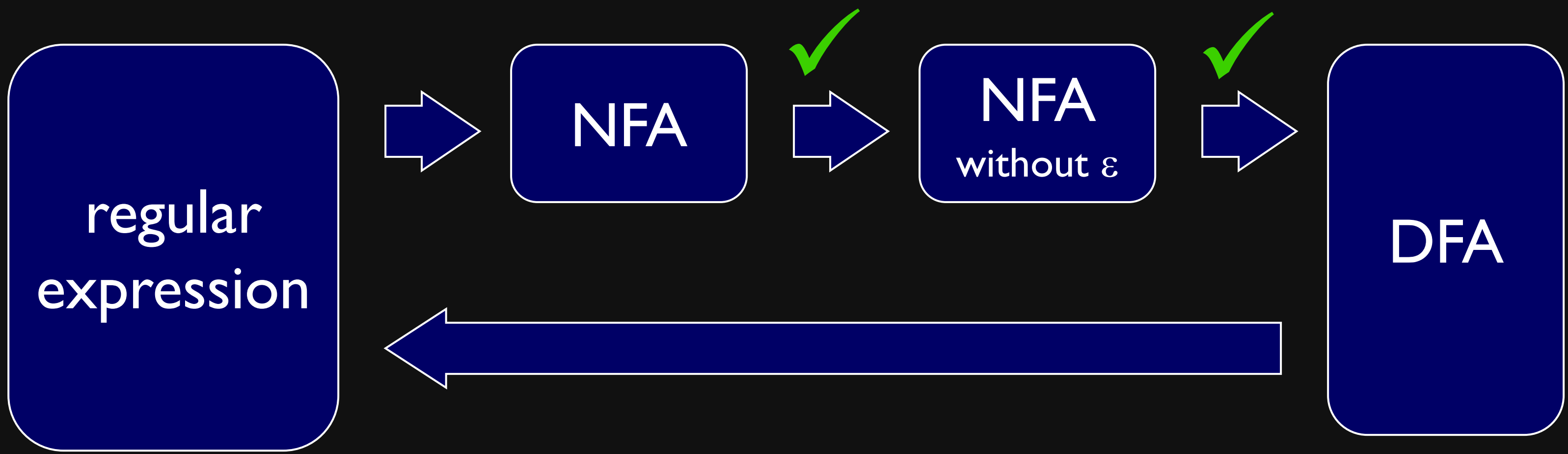


Main theorem for regular languages

A language is regular if and only if it is the language of some DFA



Road map

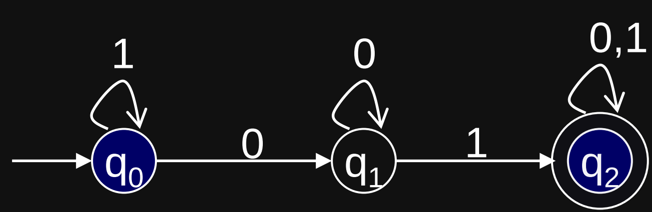


DFA to RE construction

DFA $\xrightarrow{\text{Theorem 1}}$ Reg Ex

Informally, trace all distinct paths (traversing cycles only once) from the start state to *each of the* final states and enumerate all the expressions along the way

Example:



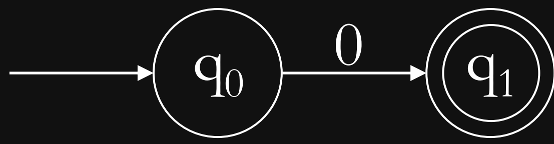
$$\underbrace{(1^*)}_{1^*}\ \underbrace{0\quad(0^*)}_{00^*}\ \underbrace{1}_{1}\ \underbrace{(0+1)^*}_{(0+1)^*}$$

$$1^*00^*1(0+1)^*$$

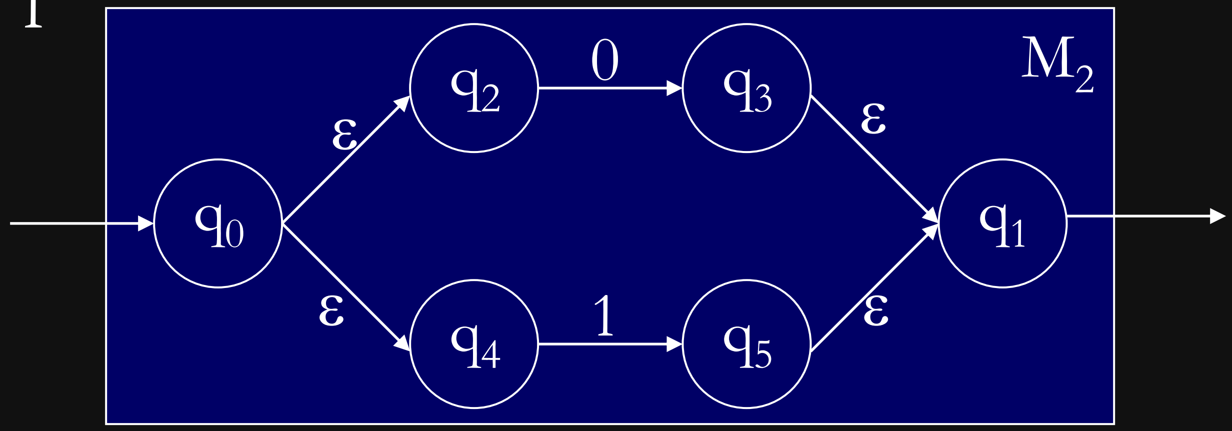
Q) What is the language?

Examples: regular expression → NFA

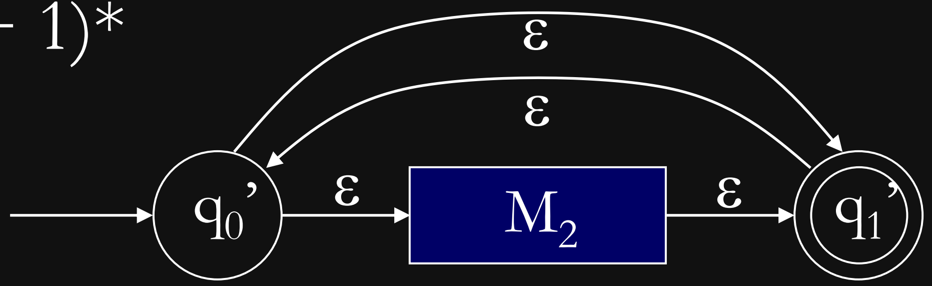
- $R_1 = 0$



- $R_2 = 0 + 1$



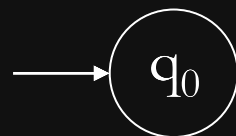
- $R_3 = (0 + 1)^*$



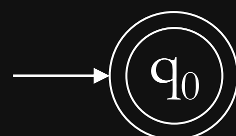
General method

regular expr → NFA

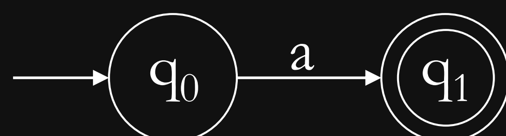
$\varnothing$



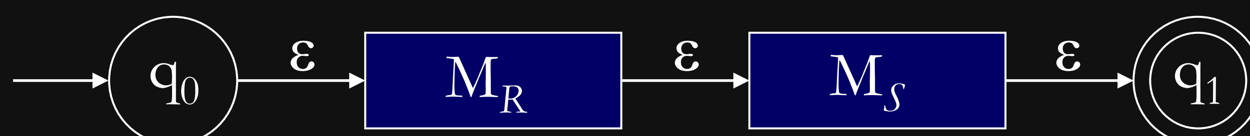
ε



symbol a



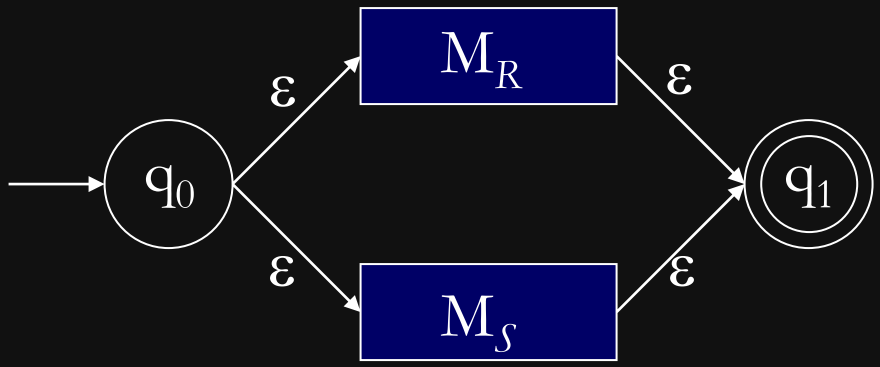
RS



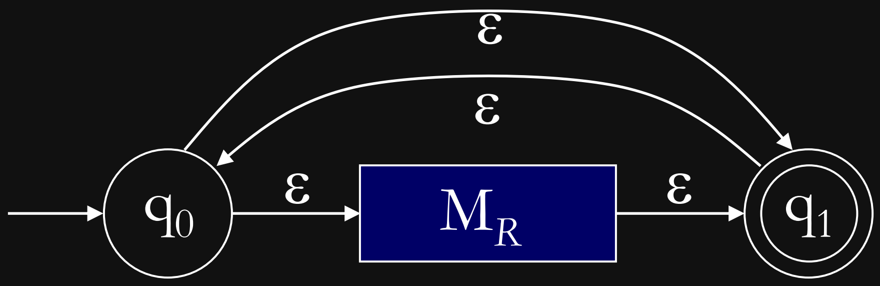
General method continued

regular expr ⇨ NFA

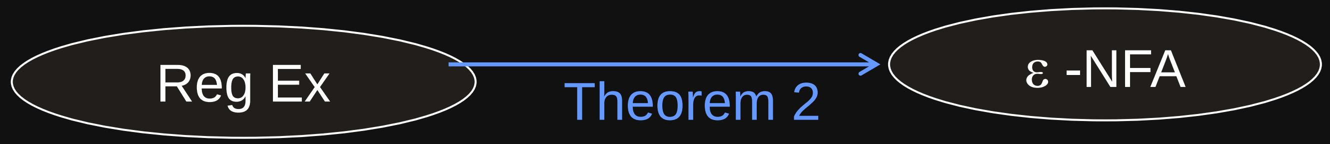
$R + S$



$R*$

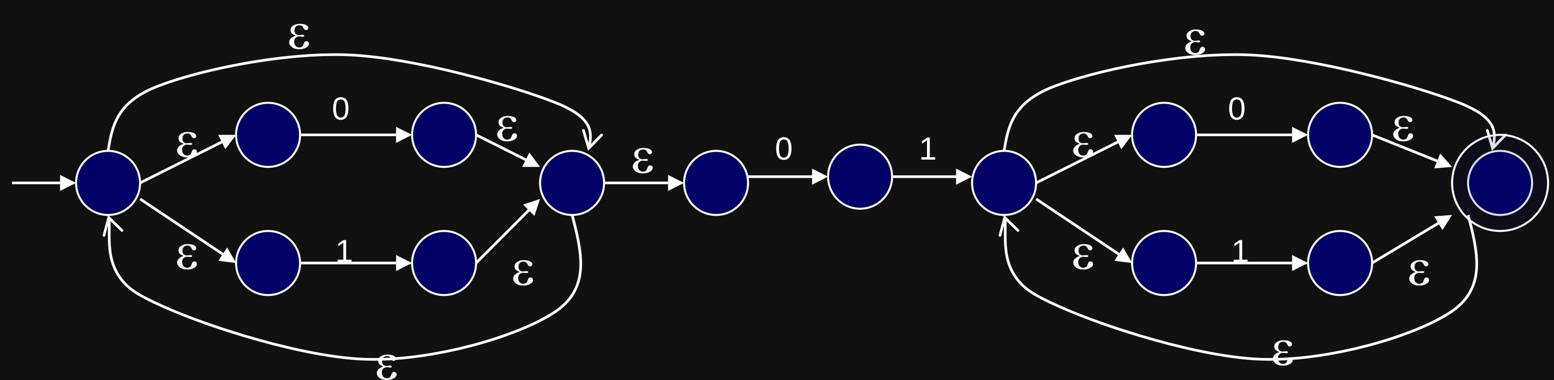


RE to ε-NFA construction



Example: (0+1)*01(0+1)*

(0+1)*   01   (0+1)*



Algebraic Laws of Regular Expressions

- Commutative:
  - $E+F = F+E$
- Associative:
  - $(E+F)+G = E+(F+G)$
  - $(EF)G = E(FG)$
- Identity:
  - $E+\Phi = E$
  - $\varepsilon E = E \varepsilon = E$
- Annihilator:
  - $\Phi E = E\Phi = \Phi$

Algebraic Laws…

- Distributive:
  - $E(F+G) = EF + EG$
  - $(F+G)E = FE+GE$
- Idempotent: $E + E = E$
- Involving Kleene closures:
  - $(E^*)^* = E^*$
  - $\Phi^* = \varepsilon$
  - $\varepsilon^* = \varepsilon$
  - $E^+ = EE^*$
  - $E? = \varepsilon + E$

True or False?

Let R and S be two regular expressions. Then:

1. $((R^*)^*)^* = R^*$?

2. $(R+S)^* = R^* + S^*$?

3. $(RS + R)^* RS = (RR^*S)^*$?