CS304: Automata and Formal Languages

Lec 8

Regular Languages, Regular Expressions, and Kleene's Theorem

Rachit Nimavat

August 14, 2025

Outline

Alphabet Σ: A finite, non-empty set of symbols

String w: A finite sequence of symbols taken from an alphabet

Language L: A set of strings over a particular alphabet

Consider $\Sigma = \{0, 1\}$ and $L = \{0\}^* \{1\} \equiv 0^*1$.

Deterministic Finite Automata (DFA)

Formal Definition

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

'symbol'	description	in our code
Q	finite set of states	$\texttt{state} \in \{0, 1, 2\}$
Σ	underlying finite alphabet	$\Sigma = \{$'0', '1'$\}$
δ	transition function between states	the 'core logic' in C code
$q_0 \in Q$	start state	$\texttt{state} = 0$
$F \subseteq Q$	accepting states	$\texttt{state} = 1$

Deterministic Finite Automata (DFA)

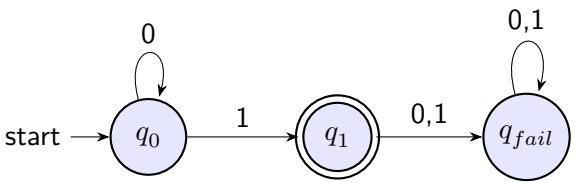
Visualizing DFA

State Diagram is the intuitive way we visualize and work with DFAs

Conventions:

- States Q are circles
- Start state (q_0) has an incoming arrow labeled start
- Accepting states (F) are double circles
- Transitions (δ) are arrows between states, labeled with input symbols from Σ

Our DFA for $L = 0^*1$:



Non-Deterministic Finite Automata (NFA)

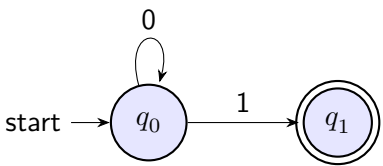
Visualizing NFA

State Diagram is the intuitive way we visualize and work with NFAs

Conventions:

- States Q are circles
- Start state (q_0) has an incoming arrow labeled start
- Accepting states (F) are double circles
- Transitions (δ) are arrows between states, labeled with input symbols from Σ

Our NFA for $L = 0^*1$:



NFA vs DFA

NFA

$N = (Q, \Sigma, \delta, q_0, F)$

$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \mapsto 2^Q$

Multiple Transitions. Possibly multiple outgoing arrows for same symbol

Zero Transitions. Possibly no outgoing arrow for a symbol (that path "dies" has license to kill)

ε-**Transitions.** Can change state without consuming an input symbol

DFA

$D = (Q, \Sigma, \delta, q_0, F)$

$\delta : Q \times \Sigma \mapsto Q$

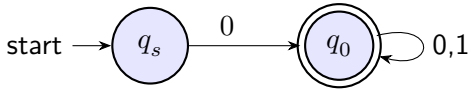
No such powers!

Theorem 1.1

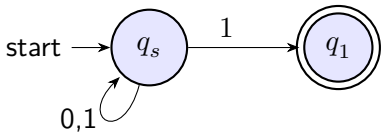
A language is recognized by an NFA if and only if it is recognized by a DFA.

Example

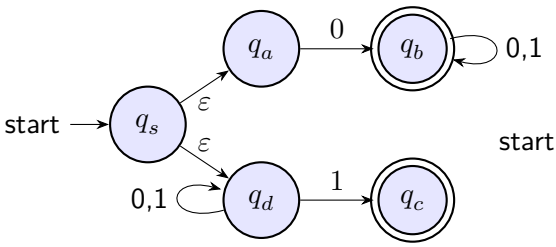
$L_1 = \{w \in \{0,1\}^* \mid w \text{ starts with } 0\}$

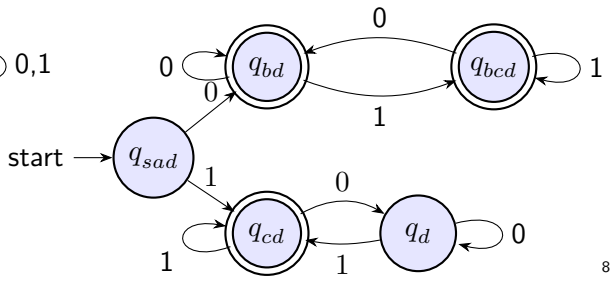


$L_2 = \{w \in \{0,1\}^* \mid w \text{ ends with } 1\}$



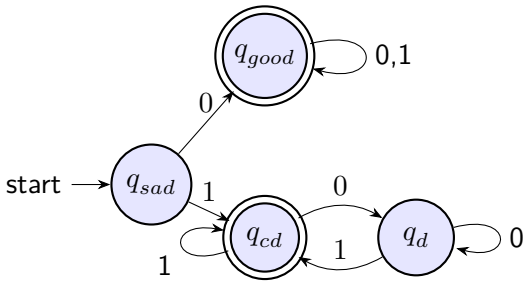
$L = L_0 \vee L_1 = \{w \in \{0,1\}^* \mid w \text{ starts with } 0 \text{ or } w \text{ ends with } 1\}.$





Example - Simplified DFA

$$L = L_0 \vee L_1 = \left\{ w \in \{0,1\}^* \mid w \text{ starts with } 0 \text{ or } w \text{ ends with } 1 \right\}.$$



What Next?

- we've learned so far to describe languages using machines: DFAs and NFAs
- they are good at recognizing strings, but are cumbersome to write
- consider find function in a text editor

  we provide it a pattern and ask to find all matching strings in the text
  we don't give it a DFA
  that 'pattern-language' is built on **Regular Expressions** (regex or RE)

Regular Expressions (RE)

Recursive Defn

A regular expression (RE) over an alphabet Σ is defined as follows:

Primitive Regular Expressions (Base case):

$\emptyset$: is an RE denoting empty language $L(\emptyset) = \{\}$

ε: is a RE the language containing only the empty string $L(\varepsilon) = \{\varepsilon\}$

a: is an RE for a symbol $a \in \Sigma$ denoting $L(a) = \{a\}$

Operations (Inductive Step): If R and S are RE, then the following are also REs:

Union ($+$ or $|$): $(R + S)$ is an RE, where $L(R + S) = L(R) \cup L(S)$

e.g. $a + b$ describes the language $\{a, b\}$

Concat ($\cdot$): (RS) is an RE, where $L(RS) = L(R)L(S)$

e.g. $(a + b)c$ describes the language $\{ac, bc\}$

Kleene Star ($*$): (R^*) is an RE, where $L(R^*) = (L(R))^*$

e.g. a^* describes the language $\{\varepsilon, a, aa, aaa, \dots, \}$

Regular Expressions

Examples

RE 0^*1 is any number of 0's followed by a 1

Language described by 0^*1: $\{1, 01, 001, 0001, \dots, \}$

RE $(a+b)^*$ is any string composed of a's and b's

Language described by $(a+b)^*$: $\{\varepsilon, a, b, ab, ba, aab, aba, baa, \dots, \}$

$(0+1)^*00(0+1)^*$ is any binary string with 00 as substring

RE $(a+\varepsilon)b$ is an optional a followed by b

Language described by $(a+\varepsilon)b$: $\{b, ab\}$

Regular Expressions

Operator Precedence

To reduce parentheses, there is a standard order of operations. The operators are evaluated in the following order, from highest to lowest precedence:

1. Kleene Star * think: power
2. Concat · think: multiplication
3. Union + or | think: addition

RE $a + bc^*$ is parsed as: $a + (b(c^*))$.

Language described by $a + bc^*$: $\{a, b, bc, bcc, bccc, \dots, \}$

RE $(a + b)c^*$ is parsed as: $(a + b)(c^*)$.

Language described by $(a + b)c^*$: $\{a, b, ac, bc, acc, bcc, \dots, \}$

Regular Language

Defn: A language L is called a regular language if and only if there exists a regular expression R that describes it: $L = L(R)$

But wait, didn't we already define regular languages earlier?
A language L is regular iff 'if and only if':

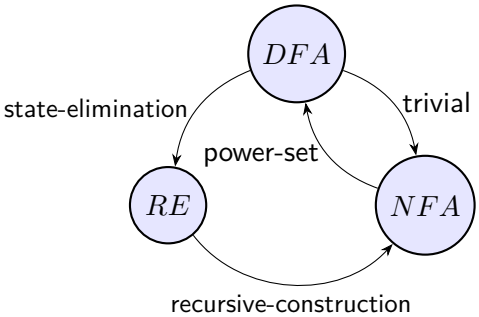
- $\exists$ an NFA A such that $L = L(A)$
- or equivalently, $\exists$ a DFA D such that $L = L(D)$

Regular Language

Theorem 3.1 (Kleene's Theorem)

All these definitions are equivalent:

- $\exists$ *a DFA* D *such that* $L = L(D)$
- $\exists$ *an NFA* A *such that* $L = L(A)$
- $\exists$ *an RE* R *such that* $L = L(R)$

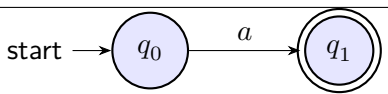
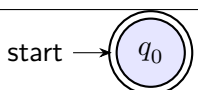
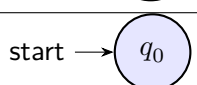


Kleene's Theorem: RE $\implies$ NFA

Step-by-step algorithm to convert an RE to an NFA

Primitive Regular Expressions (Base case):

- a (where $a \in \Sigma$)
- ϵ
- $\emptyset$

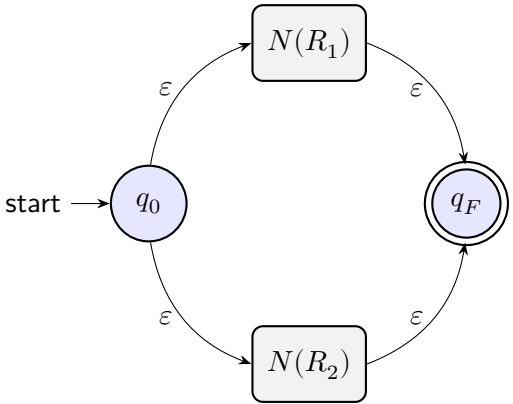
RE	NFA
a (where $a \in \Sigma$)	
ϵ	
$\emptyset$	

Kleene's Theorem: RE $\Longrightarrow$ NFA

Step-by-step algorithm to convert an RE to an NFA

Combining Machines (Inductive step):

- $R_1 + R_2$ (union)



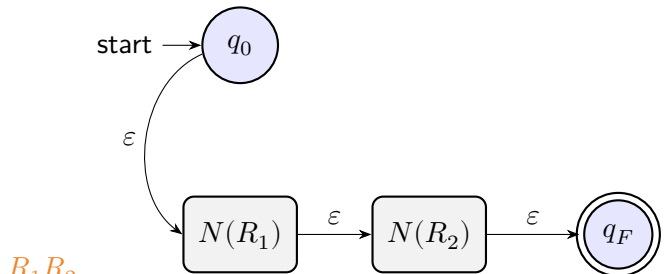
$R_1 + R_2$

Kleene's Theorem: RE $\Longrightarrow$ NFA

Step-by-step algorithm to convert an RE to an NFA

Combining Machines (Inductive step):

- $R_1 R_2$
  (concatenation)

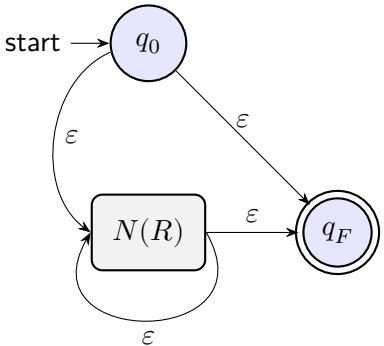


Kleene's Theorem: RE $\Longrightarrow$ NFA

Step-by-step algorithm to convert an RE to an NFA

Combining Machines (Inductive step):

- R_1^* (Kleene star)



R^*