CS304: Automata and Formal Languages

Lec 10

Pumping Lemma

Rachit Nimavat

August 22, 2025

Outline

Deterministic Finite Automata (DFA)

Formal Definition

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

'symbol'	description
Q	finite set of states
Σ	underlying finite alphabet
δ	transition function between states
$q_0 \in Q$	start state
$F \subseteq Q$	accepting states

Deterministic Finite Automata (DFA)

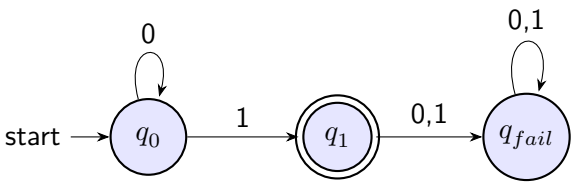
Visualizing DFA

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)

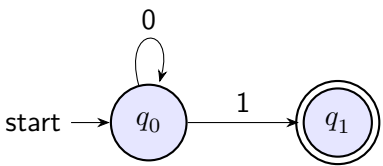
Visualizing NFA

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.

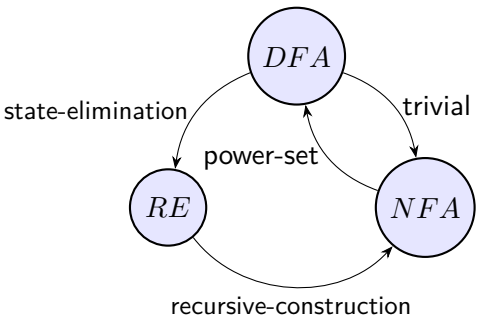
Regular Language

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

Theorem 2.1 (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* $L = L(R)$



Pigeonhole Principle

Consider a DFA M with n states.

Consider a string w of length $|w| > n$.

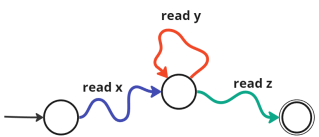
If M accepts w, it must visit at least one state multiple times while processing w

Pigeonhole Principle!

There must be a loop in the computation path

We will show that M must accept a family of related strings

Pumping Lemma



Lemma 3.1

For each regular language L, there is a constant p (the pumping length) such that any string $s \in L$ with $|s| \geq p$ can be divided into three parts, $s = xyz$, such that

i. $|y| > 0$ *("looped-string" y is not empty)*

ii. $|xy| \leq p$ *(the loop starts within the first p characters)*

iii. $\forall i \geq 0$, *the string* $xy^iz \in L$ *(we can pump the loop zero, one, or many times, and the resulting string must still be accepted)*

HW: Go over the proof of the pumping lemma in the ALC book.

How to use the Pumping Lemma?

Prove that $L = \{0^n 1^n \mid n \geq 0\}$ is not regular

Assume for contradiction that L is regular

From pumping lemma, there is a constant p with "looping property"

Let's pick a string $s = 0^p 1^p$. Notice that $s \in L$

Consider its decomposition $s = xyz$ given by the pumping lemma

$|xy| \leq p \implies y = 0^{|y|}$ y consists of all zeroes

Choose $i = 2$. From pumping lemma, $s_2 = xy^2z = xy \cdot y \cdot z = 0^{p+|y|}1^p \in L$

Contradiction! since $|y| > 0$)

Example 2

Prove that $L = \{w \in \{a,b\}^* \mid w$ is a palindrome$\}$ is irregular

Assume for contradiction that L is regular

From pumping lemma, there is a constant p with "looping property"

Let's pick a string $s = a^p b a^p$. Notice that $s \in L$

Consider its decomposition $s = xyz$ given by the pumping lemma

$|xy| \leq p \implies y = a^{|y|}$ and $|y| > 0$ y is non-empty and consists of all as

Choose $i = 0$. From pumping lemma, $s_0 = xz = a^{p-|y|} b a^p$, which is not a palindrome since $|y| > 0$

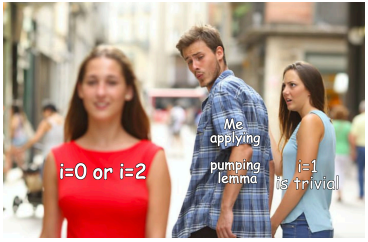
Contradiction!

"The Rule"

To prove that a language L is irregular,

- Assume for contradiction that L is regular and M is a DFA accepting it
- The pumping lemma gives us a constant p
- Cleverly choose a string $w \in L$ with $|w| \geq p$
- The pumping lemma gives us the decomposition $s = xyz$ with guarantees (note: we cannot choose x, y, z they are given by pumping lemma)
- Cleverly choose $i \geq 0$ so that the pumped string $s_i = xy^iz \notin L$.
- But M must accept s_i

Contradiction!



Summary So Far

Regular languages are limited by their finite memory (states)

Pumping Lemma formalizes this by showing that any sufficiently long string in a regular language must contain a "pumpable" loop

Can use this to prove a language is irregular by finding a string that breaks this pumping property.

What's Next?

We've hit the ceiling on the power of finite automata

To recognize more complex languages like 0^n1^n, we need a more powerful model of computation that includes **memory**

Context-Free Grammars and **Pushdown Automata**