COMS W3261 – Theory of Computation

Lecture 2. Regular Operations & Nondeterminism

twrand.github.io/3261.html.

Announcements!

HW #1 up (on website)
Due Tuesday, 7/6/2021 @ 11:59 PM EST
Note: Skip problem 2.1

Today:

1. Quick Review
2. Regular Operations
3. Regular languages are closed under union.

---

4. Nondeterministic Finite Automata (NFAs)
5. Proof that any NFA can be converted into a DFA that recognizes the same language.

1. Review

Last time:

- Languages are sets of strings ≈ concepts
- Deterministic Finite Automata (DFAs)

specify a procedure for deciding whether or not a string w is in a language L.

- Set of recognized strings: $L(D)$
  (language recognized by D.)

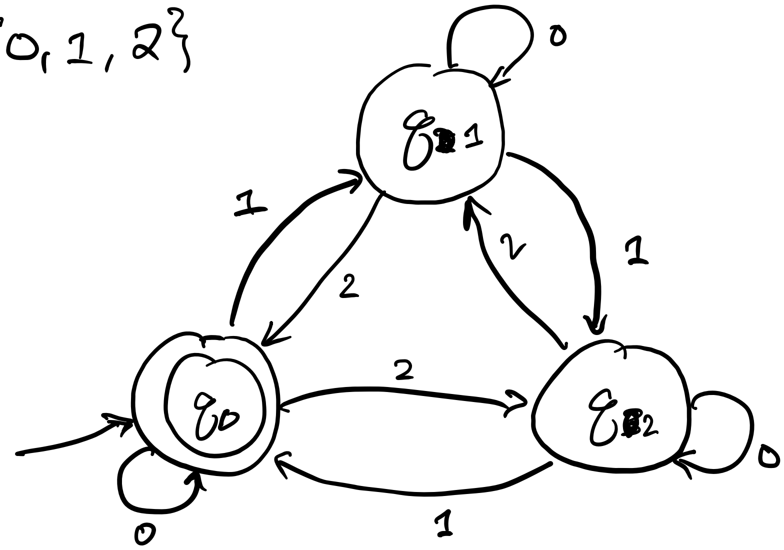
- Regular languages are those recognized by some DFA

DFA was a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

states, alphabet, transition function, start state, set of accept states.

State diagram: contains all the same information.

Example. On $\Sigma = \{0, 1, 2\}$



recognized: $\{ w \mid \Sigma \text{ digits of } w \equiv 0 \pmod 3 \}$

0 1 1 2 1 2 ...

2. Regular Operations

Idea: regular language $\longleftrightarrow$ recognized by some DFA.

It would be nice to be able to say

'If A and B are regular, $A \cup B$ is regular'

Def. (Some regular operations.)

Union: $A \cup B := \{ x \mid x \in A \text{ or } x \in B \}$

Concatenation: $A \cdot B := \{ xy \mid x \in A, y \in B \}$

(Kleene) Star: $A^* := \{ x_1 x_2 \cdots x_k \mid k \geq 0, x_i \in A \}$

Example. $A = \{\text{red, blue}\}$, $B = \{\text{cat, dog}\}$

$A \cup B = \{\text{red, blue, cat, dog}\}$

$A \circ B = \{\text{redcat, reddog, bluecat, bluedog}\}$

$A^* = \{\underbrace{\varepsilon}_{k=0}, \underbrace{\text{red, blue}}_{k=1}, \underbrace{\text{redred, redblue, bluered, blueblue}}_{2}, \text{redbluered} \ldots\}$

$\{\varepsilon\}^* = \{\varepsilon\}$

$\emptyset^* = \{\}^* = \{\}$

Theorem. Regular Languages are closed under union $\cup$.

(Equiv: If A regular, B regular, then $A \cup B$ is regular.)

(Equiv: If A, B are both recognized by some DFA, then there exists a DFA that recognizes $A \cup B$.)

Idea: Simulate running M_1 for A and M_2 for B at the same time. Accept if either accepts. Our new machine will use a pair of states from M_1 and M_2 as a single state.

Proof. M_1 is a DFA $(Q_1, \Sigma, \delta_1, q_1, F)$ recognizing A.

M_2 is a DFA $(Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizing B.

We'll build a new machine $M = (Q, \Sigma, \delta, q_0, F)$ and show it recognizes $A \cup B$.

$Q = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\}$

Σ same.

$q_0 = (q_1, q_2)$

and? $\longrightarrow A \cap B$

$F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ OR } r_2 \in F_2\}$

input symbol

δ: For each $(r_1, r_2) \in Q$ and for each symbol $a \in \Sigma$
$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)). \quad \square$$

(Imagine putting some string w into M. The two components of the state update "independently." When I stop, accept if at least one simulation accepts — if $w \in A$ or $w \in B$.)

Theorem: Regular languages are closed under concatenation ($\circ$).

New ingredient: NONDETERMINISM.

Deterministic Finite Automata.

next step in the computation is completely determined.

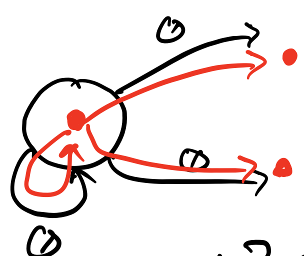
$$\delta : Q \times \Sigma \to Q$$

Idea: what if we break determinism?

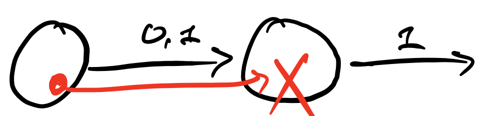
New rules for (Non)deterministic Finite Automata (NFAs).

1. Multiple out-arrows/transitions for one symbol? Split into two branches and take all.

Read in a 0:

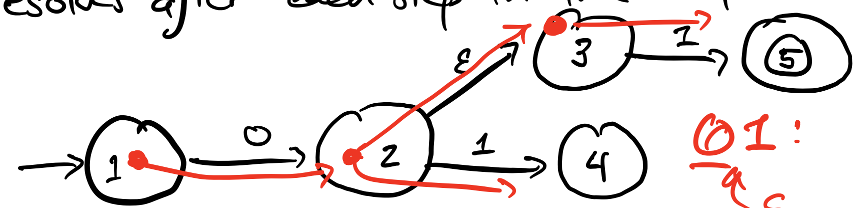


2. No transitions for a symbol? The branch "dies."
All branches of computation die: reject.



Read in 00:

3. ε-arrow/ε-transition indicates an extra "free branch" that resolves after each step in the computation.



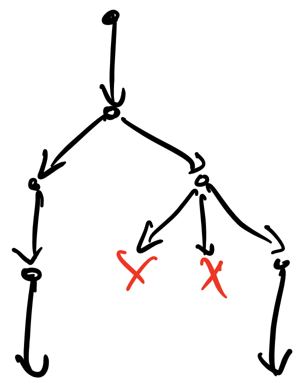
01: ε

4. Accept if any live branch accepts at the end of the input string.

Determinism

accept
(reject)

Nondeterminism



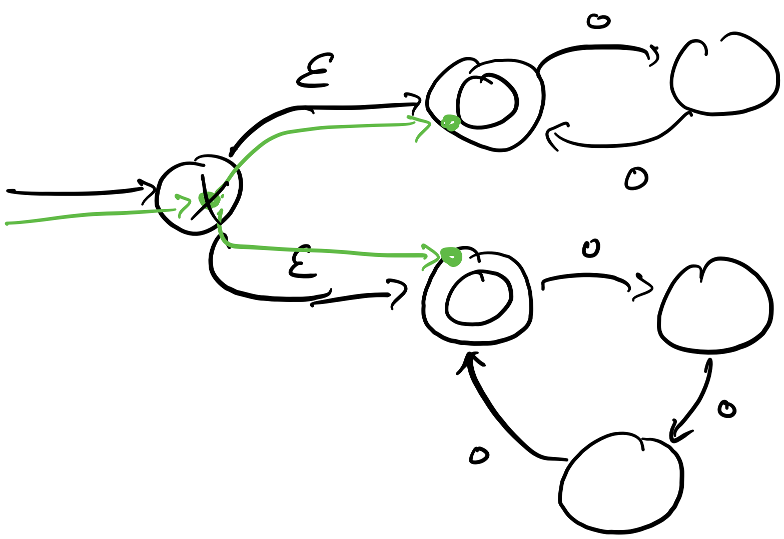
reject

accept

Example 1. (NFA state diagram, on $\Sigma = \{0\}$)

Goal: Recognize language

$\{w \mid |w| \text{ is divisible by } 2 \text{ or by } 3\}$

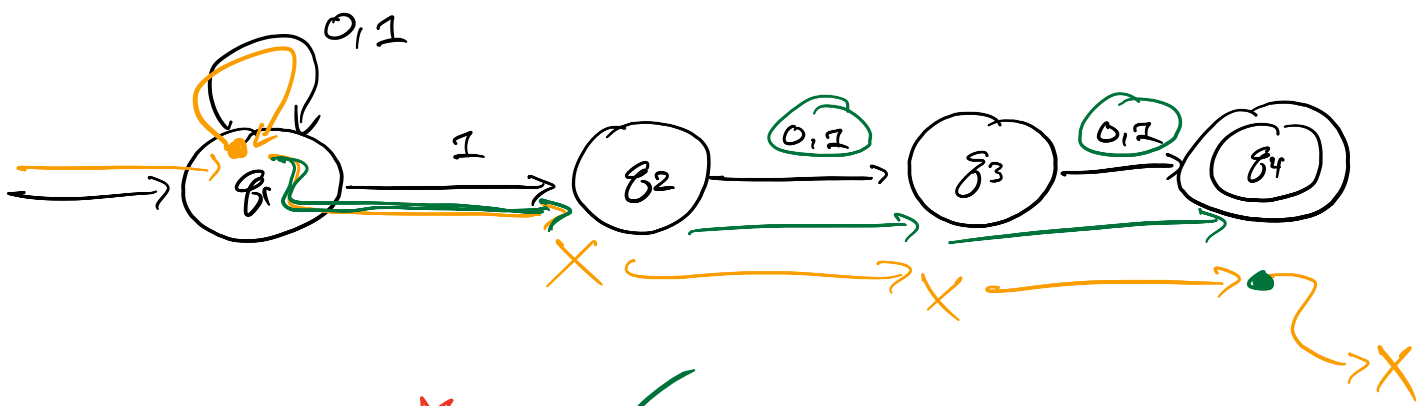


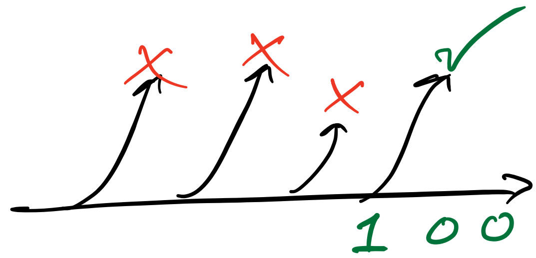
// accepts $|w|$ divisible by 2.

// accepts $|w|$ divisible by 3.

Example 2. Over the alphabet $\Sigma = \{0, 1\}$:

Goal: recognizes $\{ w \mid w$ has a '1' in the third-to-last place.$\}$





Def. (Power set.) The power set of Q is denoted $\mathcal{P}(Q)$ and is the set of all subsets of Q.

$$Q = \{a, b\}. \quad \mathcal{P}(Q) = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$$

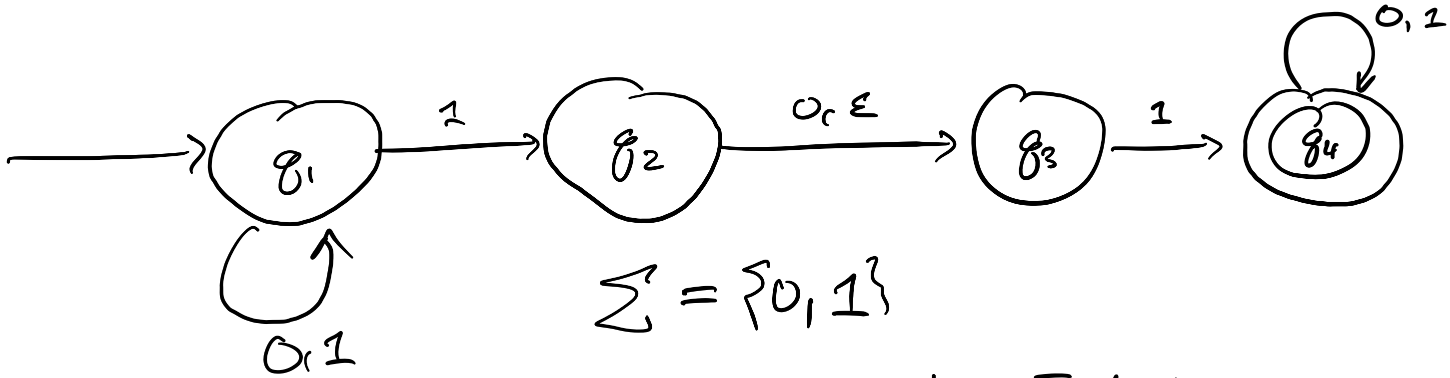
Def. (NFA, formally.) Let $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$. An NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:

- Q is a finite set of states,
- Σ is a finite alphabet,
- q_0 is a start state,
- $F \subseteq Q$ is the set of accept states,
- $\boxed{\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)}$

An NFA N accepts a string $w = w_1 w_2 \cdots w_m$, where each $w_i \in \Sigma_\varepsilon$, if $\exists$ a sequence of states $r_0, r_1 \cdots r_m \in Q$ s.t.

- $r_0 = q_0$
- $r_{i+1} \in \delta(r_i, w_{i+1})$ for $i = 0, 1, \cdots m-1$
- $r_m \in F$.

Example: Writing formal def'n of an NFA state diagram.



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

This state diagram corresponds to the 5-tuple

$M = (Q, \Sigma, \delta, q_0, F)$, where

$Q = \{q_1, q_2, q_3, q_4\}$

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

$q_0 = q_1$

$F = \{q_4\}$

$\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$

δ:

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

Next: Converting any NFA to a DFA.

Pause: back at 11:45.

Idea: Show every NFA $\longrightarrow$ DFA.

This will imply:

Fact. A language is regular if and only if it is recognized by some NFA.

(reg $\longrightarrow$ DFA $\longrightarrow$ NFA)

(NFA $\longrightarrow$ DFA $\longrightarrow$ reg)

(NFA ⟶ DFA ⟶ reg)

Theorem. Every NFA corresponds to a DFA that recognizes the same language.

Strategy. Our DFA will use every possible set of states in the NFA as a state. Our transition function will then simulate all live branches of the NFA.

Proof: Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA that recognizes the language A. We'll build a DFA that recognizes A. Build DFA $M = (Q', \Sigma, \delta', q_0', F')$

- $Q' = \mathcal{P}(Q)$
- Σ same
- $F' = \{R \in Q' \mid R \text{ contains an accept state for } N.\}$

Recall: at each step of our NFA,

1. We start at some set R of states
2. We follow all transitions corresponding to the next input symbol a.
3. We follow (and don't follow) all ε-arrows.

Define: $\delta'(R, a)$, for any set $R \subseteq Q$, and any $a \in \Sigma$.

[For $R \subseteq Q$, let $E(R)$ denote all states in Q reachable by ε-arrows from R.]

- $\delta'(R, a) = \underbrace{\{q \in Q \mid q \in E(\delta(r,a)), r \in R\}}_{\subseteq Q}$

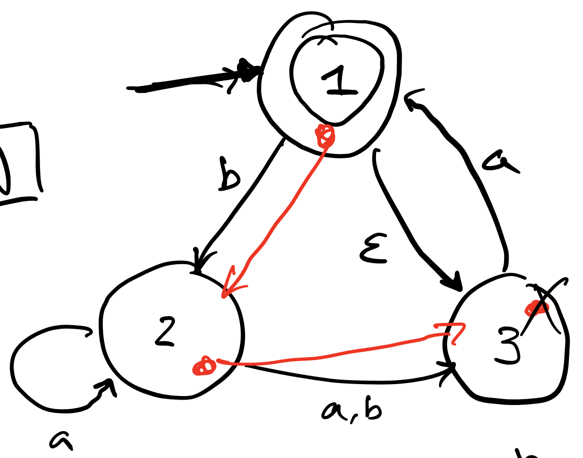
- $q_0' = E(\{q_0\})$. $\square$

Imagine running our new DFA M on a string w. We start at $E(\{q_0\})$. We simulate the NFA, and then accept whenever our simulation accepts.

accept whenever ...

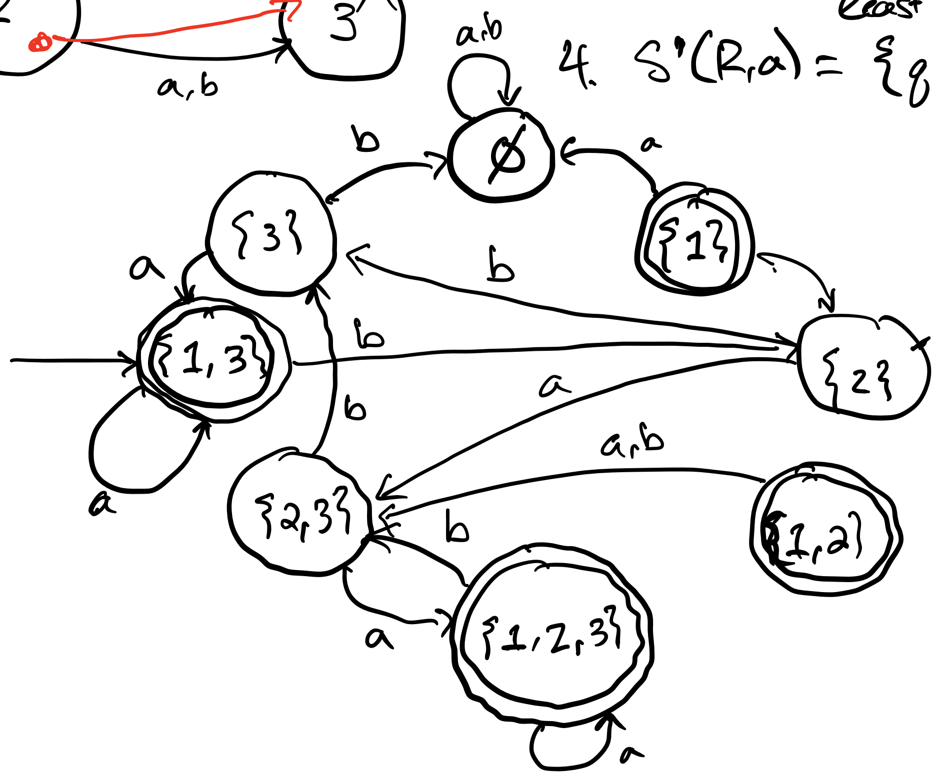
Example: Converting an NFA to a DFA. $\Sigma = \{a, b\}$.

$\boxed{N}$



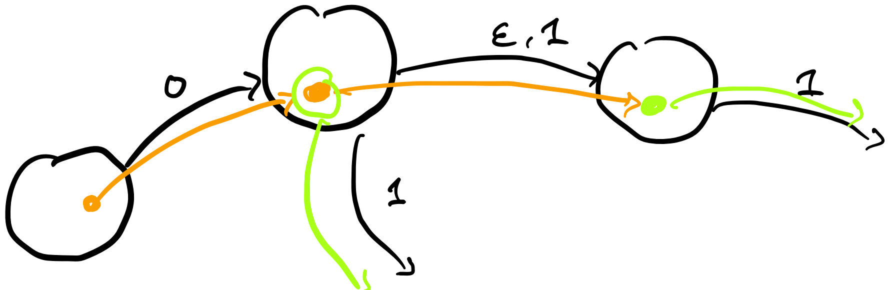
$M = (Q', \Sigma, \delta', q_0', F')$

1. $Q' = \mathcal{P}(Q)$
2. $q_0' = E(\{q_0\})$
3. $F' = \{R \in Q' \mid R$ contains at least 1 accept state of $N\}$
4. $\delta'(R,a) = \{q \in Q \mid q \in E(\delta(r,a))$ for some $r \in R\}$



Read in:

0, 1

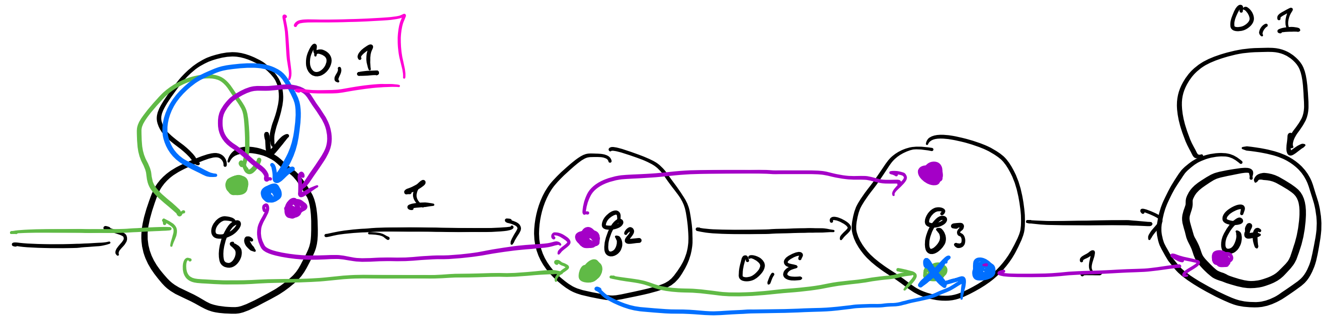


After reading 0:

see an ε-edge

add an extra branch that follows it

Step-by-step NFA evaluation. $\Sigma = \{0, 1\}$



Evaluate execution on 101. ✓ 0100101

3 live

2 live

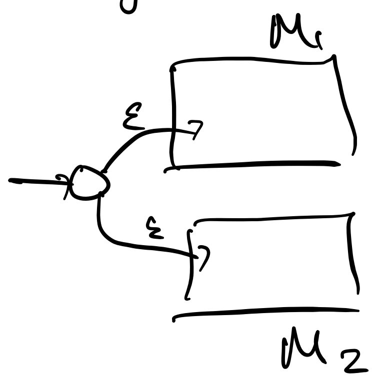
4 live branches!

$\{w \mid w$ has '11' or an '101' substring$\}$

Next time:

Closure of regular languages under regular operations using NFAs!

A regular, B regular $\rightarrow$ $A \cup B$



Reading: Second part 1.1 in Sipser, part. 1.2.

HW due Tuesday, 7/6, at 11:59PM.

(SKIP question 2.1).