COMS W3261: Theory of Computation

Lecture 2: Regular operations & nondeterminism

twrand.github.io/3261.html

Announcements: HW #1 up!

Due Tuesday, 7/6/21, 11:59 PM

Today:

1. Review
2. Regular operations
3. Prove that regular languages are closed under union
4. Nondeterministic Finite Automaton (NFA)
5. Prove that any NFA can be converted into a DFA that recognizes the same language.

1. Review

Last time: Languages are sets of strings. Languages $\approx$ math concepts

- DFAs specify a procedure for deciding **whether** or **not** a certain input string is in a language.
- Set of strings recognized by a DFA D, $L(D)$, is the language **recognized** by D.
- **Regular languages** := those recognized by some DFA.

Def. A DFA is formally written as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

where Q is a finite set of states,

Σ is a finite alphabet of symbols,

Σ is a finite alphabet of symbols,

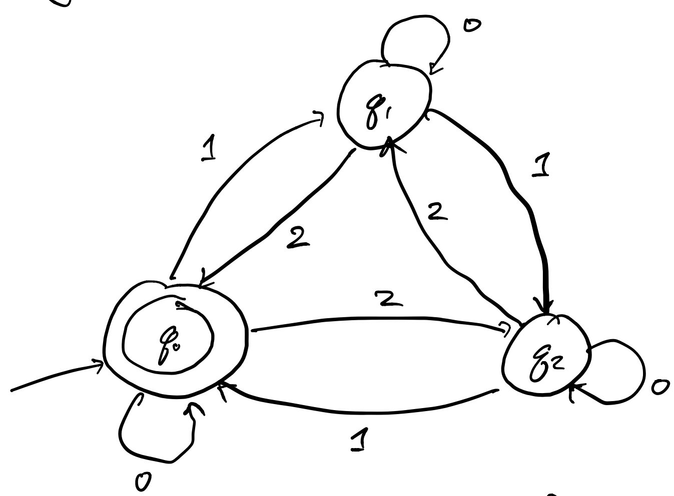
$\delta: Q \times \Sigma \to Q$, a transition function that takes a (state, symbol) pair and gives the next state,

q_0 is the start state, and

$F \subseteq Q$ is a set of accept states.

A state diagram contains all the same information as the 5-tuple.

E.g., on $\Sigma = \{0, 1, 2\}$,



recognizes $\{w \mid \text{the sum of the digits of } w \equiv 0 \pmod 3\}$

---

2. Regular Operations

Idea: regular languages := those recognized by a DFA.

how to prove this? 1) Show a DFA that recognizes a given language.

Example: It would be nice to say things like

'If A and B are regular, C is also regular.'

Def. (Some regular operators.)

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

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

(not the same as $B \circ A$).

(Kleene) Star: $A^* := \{x_1 x_2 x_3 \cdots x_k \mid k \geq 0, x_i \in A\}$

<u>Example:</u> $A = \{\text{red}, \text{blue}\}$, $B = \{\text{cat}, \text{dog}\}$

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

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

$A^* = \{\varepsilon, \text{red}, \text{blue}, \text{redred}, \text{redblue}, \text{blueblue}, \text{bluered}, \text{redredred} \ldots\}$

// $\{\varepsilon\}^* = \{\varepsilon\}$ $\quad$ $\{\}^* = \{\}^*$ (aside.)

<u>Theorem.</u> Regular languages are closed under union.

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

(If A is recognized by a DFA M_1, and B is recognized by a DFA M_2, then $A \cup B$ is recognized by a DFA M.)

<u>Idea:</u> Simulate M_1 and M_2 at the same time, and accept if either simulated machine accepts.

<u>Proof.</u> Let M_1 be a DFA $(Q_1, \Sigma, \delta_1, q_1, F_1)$ that recognizes A.
Let M_2 be a DFA $(Q_2, \Sigma, \delta_2, q_2, F_2)$ that recognizes B.

(A, B regular languages.)

We want to build a machine $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $A \cup B$.

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

$$\Sigma = \Sigma.$$

δ: For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$,

δ: For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$,

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$

$$q_0 = (q_1, q_2)$$

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

Why does this DFA recognize $A \cup B$? Imagine an input string $w = w_1 w_2 \ldots w_n$. On each symbol, each component of our state updates "independently" to simulate M_1 or M_2. When we stop, accept if at least one of M_1, M_2 accepts. $\square$

<u>Theorem</u>: Regular languages are closed under concatenation.

New ingredient: NONDETERMINISM.