Regular Expressions

Recall: NFAs and DFAs both recognize the class of regular languages

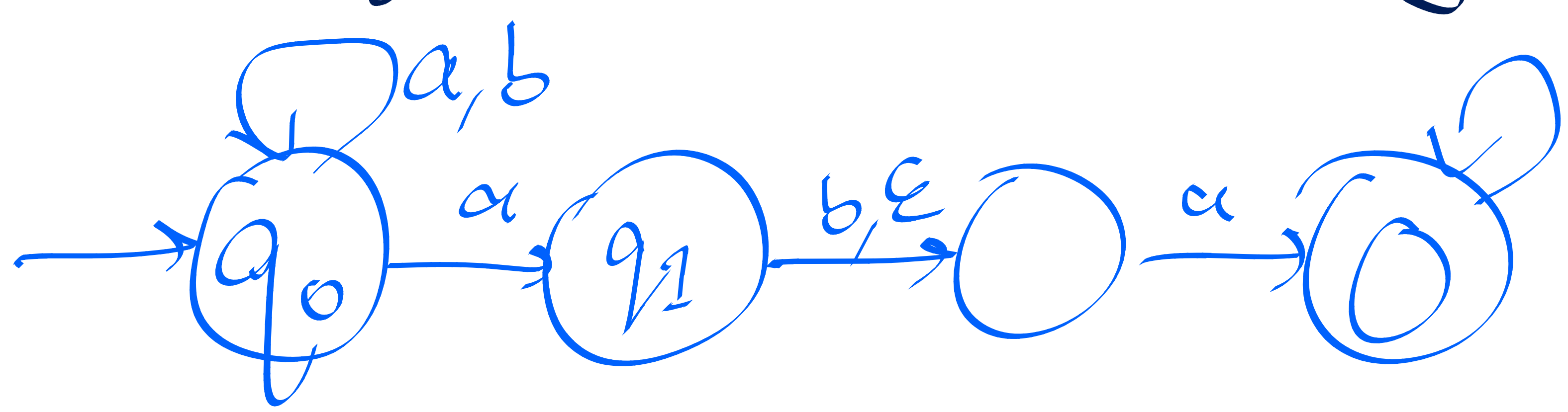
Today: Another way to specify regular languages

When we move from an NFA to a DFA, there is potentially an exponential blow-up in the number of states.

Consider, over $\Sigma = \{a, b\}$, the following language

$\mathcal{L} = \{ s \mid s \text{ contains aba or aa as a substring} \}$

What is a finite automaton recognizing $\mathcal{L}$?



But what if I have to ask my text editor to find all such words?

Is there a less informal way to do it? Regular expressions

What are the basic regular expressions?

- a, for every $a \in \Sigma$
- ε
- $\emptyset$

Suppose φ and ψ are regular expressions. Then, so are:

- $\varphi \cdot \psi$
- $\varphi + \psi$
- φ^*
- $\sim \varphi$

every word avoids φ

every word matches some iterations of φ

every word either matches φ or matches ψ

every word is of the form $\omega_1 \omega_2$, where ω_1 matches φ, ω_2 matches ψ

$\mathcal{L} = \{ s \mid s$ contains aba or aa as a substring $\}$

What is a regular expression that represents $\mathcal{L}$?

$\Sigma = \{a, b\}$

$$(a+b)^* aba (a+b)^* + (a+b)^* aa (a+b)^*$$

$$\Sigma^* aba \Sigma^* + \Sigma^* aa \Sigma^*$$

Is there a connection between this and the earlier automaton?

L : strings over $\Sigma = \{a, b\}$ with an odd number of a s

$\mathcal{L}$ = strings over $\Sigma = \{a, b\}$ ending in b and not containing aa

$$\sim\left(\sim\left(\Sigma^* b\right) + \Sigma^* aa \Sigma^*\right)$$

Reg is the class of languages expressible as regexes

We know Reg is the class recognized by finite-state automata

So we show that regexes are "equivalent" to FSAs.

For each regex expressing $\mathcal{L}$, there is an NFA M s.t. $\mathcal{L}(M) = \mathcal{L}$.

We have automata accepting each of the regex patterns

Single letter, empty string, empty language

Union $(+)$, concatenation $(\cdot)$, star $(*)$, complement $(\sim)$

Proof by induction On what?

For each NFA M, there exists a regex representing $\mathcal{L}(M)$.

Basic idea: Keep track of the patterns tracked by each accepting path

Enumerate the states of M. $\{1, \ldots, n\}$ initial state

Maintain a set $\mathcal{L}_{ij}^{k}$ of all strings that take M from i to j, s.t. any state (other than i & j) encountered on this path is $\leq k$.

$$\mathcal{L}_{ij}^{0} = \begin{cases} \{a \mid \delta(i,a) = j\} & i \neq j \\ \{a \mid \delta(i,a) = j\} \cup \{\varepsilon\} & i = j \end{cases}$$

$$\mathcal{L}_{ij}^{k} = \mathcal{L}_{ij}^{k-1} \cup \mathcal{L}_{ik}^{k-1} \cdot \left(\mathcal{L}_{kk}^{k-1}\right)^{*} \cdot \mathcal{L}_{kj}^{k-1}$$

$$\mathcal{L}(M) = \bigcup_{s \in F} \mathcal{L}_{1s}^{n}$$