In Computer Science, we operationalize "hardest" as "requires most resources", where resources might be memory, time, parallelism, randomness, power, etc. To be able to compare "hardness" of problems, we use a consistent description of problems

Problem ~ Language.

Input: String

Output: Yes/ No, where Yes means that the input string matches the pattern or property described by the problem.

So far: we saw that regular expressions are convenient ways of describring patterns in strings. **Finite automata** give a model of computation for processing strings and and classifying them into Yes (accepted) or No (rejected). We will see that each set of strings is described by a regular expression if and only if there is a FA that recognizes it. Another way of thinking about it: properties described by regular expressions require exactly the computational power of these finite automata.

Wednesday: Finite automaton constructions

Review: Formal definition of finite automaton: $M = (Q, \Sigma, \delta, q_0, F)$

- Finite set of states Q (memory)
- Alphabet Σ
- Transition function δ $Q \times \Sigma \to Q$
- Start state q_0
- Accept (final) states F

Can represent M with state diagram : graph w/ lots of decoration

In the state diagram of M, how many outgoing arrows are there from each state? $|\Sigma|$

$M = (\{q, r, s\}, \{a, b\}, \delta, q, \{q\})$ where δ is (rows labelled by states and columns labelled by symbols):

formal def

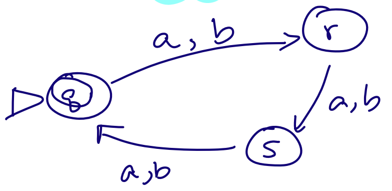
transition function

Σ

δ	a	b
q	r	r
r	s	s
s	q	q

Q

The state diagram for M is



Give two examples of strings that are accepted by M and two examples of strings that are rejected by M:

M accepts abb and M accepts ε, bbb, abbaab
M rejects a and ab.

$L(M) =$ $\{ w \in \Sigma^* \mid w$ has length an integer multiple of 3 $\}$.

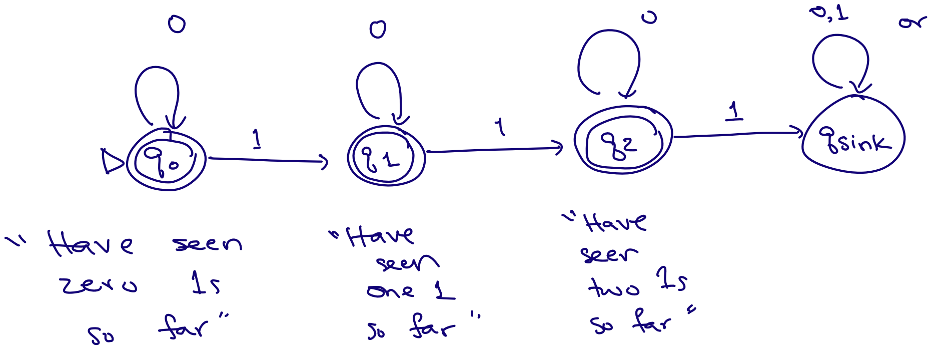
A regular expression describing $L(M)$ is $((a \cup b)(a \cup b)(a \cup b))^*$

observe: don't care about 0s.

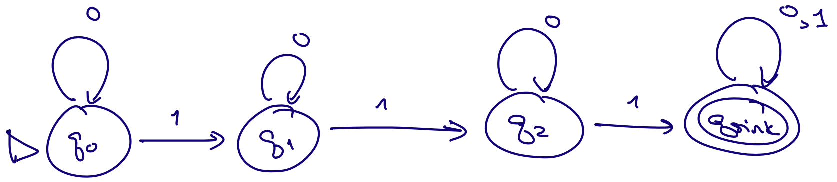
Let the alphabet be $\Sigma_1 = \{0, 1\}$.

A state diagram for a finite automaton that recognizes $\{w \in \Sigma_1^* \mid w \text{ contains at most two 1's}\}$ is

i.e. zero 1s
or one 1
or two 1s.



A state diagram for a finite automaton that recognizes $\{w \in \Sigma_1^* \mid w \text{ contains more than two 1's}\}$ is



Strategy: Add "labels" for states in the state diagram, e.g. "have not seen any of desired pattern yet" or "sink state". Then, we can use the analysis of the roles of the states in the state diagram to work towards a description of the language recognized by the finite automaton.

English

Justify? $L(M) = \boxed{\quad\quad\quad\quad}$

Need to prove $\subseteq$ and $\supseteq$

When we say $x \in L(M)$, we mean computation of M or x lands in a state in set of accept states.

A useful bit of terminology: the **iterated transition function** of a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ is defined recursively by

$$\delta^*(\ (q, w)\ ) = \begin{cases} q & \text{if } q \in Q, w = \varepsilon \\ \delta(\ (q, a)\ ) & \text{if } q \in Q,\ w = a \in \Sigma \\ \delta(\ (\delta^*(q, u), a)\ ) & \text{if } q \in Q,\ w = ua \text{ where } u \in \Sigma^* \text{ and } a \in \Sigma \end{cases}$$

Using this terminology, M accepts a string w over Σ if and only if $\delta^*(\ (q_0, w)\ ) \in F$.

Suppose A is a language over an alphabet Σ. By definition, this means A is a subset of Σ^*. **Claim:** if there is a DFA M such that $L(M) = A$ then there is another DFA, let's call it M', such that $L(M') = \overline{A}$, the complement of A, defined as $\{w \in \Sigma^* \mid w \notin A\}$.

Proof idea: Accept strings that would have been rejected and vice versa i.e. same graph, opposite ◎/○.

Proof:

Let A be a language over an alphabet Σ.

Assume there is a DFA M s.t. $L(M) = A$.

i.e. there is $M = (Q, \Sigma, \delta, q_0, F)$ and for each $x \in A$, M accepts x and for each $x \notin A$ M rejects x

Want to show (WTS) that there is another DFA, M', such that $L(M') = \overline{A} = \{w \in \Sigma^* \mid w \notin A\}$.

Need to build M', by giving its formal definition.

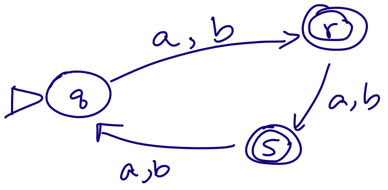
Define $M' = (\underbrace{Q, \Sigma, \delta, q_0}_{\text{Same as for M!}}, \{q \in Q \mid q \notin F\})$

PROOF CONTINUES NEXT PAGE.

WTS $L(M') = \overline{A}$

Application: Design a finite automaton that recognizes the language of all strings over $\{a, b\}$ whose length is not a multiple of 3.

Notice: this is the complement of the set of string over $\{a,b\}$ whose length <u>is</u> a multiple of 3. Thus applying the theorem to the example from the start of class we get:



Note: On Friday, we'll see a new kind of finite automaton. It will be helpful to distinguish it from the machines we've been talking about so we'll use **Deterministic Finite Automaton** (DFA) to refer to the machines from Section 1.1.

Proof cont'd:

Have $M = (Q, \Sigma, \delta, q_0, F)$ $L(M) = A$ and

$M' = (Q, \Sigma, \delta, q_0, \{q \in Q \mid q \notin F\})$

Want to show $L(M') = \{w \in \Sigma^* \mid w \notin A\}$.

Goal ① Consider arbitrary $x \in L(M')$. WTS $x \in \{w \in \Sigma^* \mid w \notin A\}$.

Using iterated transition function and definition of M' and $L(M')$,

$$\delta^*(q_0, x) \in \underbrace{\{q \in Q \mid q \notin F\}}_{\text{the set of accept states of } M'}$$

which means $\delta^*(q_0, x) \notin F$ and since δ is also the transition function of M and q_0 is the start state of M, we get that M rejects x. Since M accepts all only string in A (b/c $L(M) = A$), this means $x \notin A$, as required.

Goal ② Consider string x with $x \notin A$. WTS $x \in L(M')$, i.e. M' accepts x.

[similar to before... left as exercise].

Note: visiting instructor on Friday

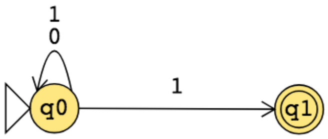
Friday: Nondeterministic automata

Nondeterministic finite automaton (Sipser Page 53) Given as $M = (Q, \Sigma, \delta, q_0, F)$

Finite set of states Q	Can be labelled by any collection of distinct names. Default: $q0, q1, \ldots$
Alphabet Σ	Each input to the automaton is a string over Σ.
Arrow labels Σ_ε	$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$. Arrows in the state diagram are labelled either by symbols from Σ or by ε
Transition function δ	$\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ gives the **set of possible next states** for a transition from the current state upon reading a symbol or spontaneously moving.
Start state q_0	Element of Q. Each computation of the machine starts at the start state.
Accept (final) states F	$F \subseteq Q$.

M accepts the input string $w \in \Sigma^*$ if and only if **there is** a computation of M on w that processes the whole string and ends in an accept state.

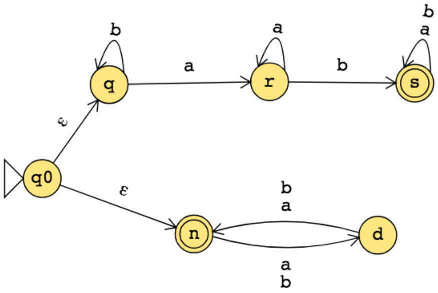
The formal definition of the NFA over $\{0, 1\}$ given by this state diagram is:



The language over $\{0, 1\}$ recognized by this NFA is:

Change the transition function to get a different NFA which accepts the empty string (and potentially other strings too).

The state diagram of an NFA over $\{a, b\}$ is below. The formal definition of this NFA is:



The language recognized by this NFA is:

Week 2 at a glance

Textbook reading: Section 1.1, 1.2

For Wednesday: Pages 41-43 (Figures 1.18, 1.19, 1.20) (examples of automata and languages).

For Friday: Pages 48-50 (Figures 1.27, 1.29) (introduction to nondeterminism).

Note: visiting instructor on Friday

For Week 3 Monday: Pages 60-61 Theorem 1.47 and Theorem 1.48 (closure proofs).

Make sure you can:

- Use regular expressions and relate them to languages and automata
  - Write and debug regular expressions using correct syntax
- Use precise notation to formally define the state diagram of DFA, NFA and use clear English to describe computations of DFA, NFA informally.
  - Design an automaton that recognizes a given language
  - Specify a general construction for DFA based on parameters
  - Design general constructions for DFA
  - Motivate the use of nondeterminism
  - Trace the computation(s) of a nondeterministic finite automaton

TODO:

#FinAid Assignment on Canvas https://canvas.ucsd.edu/courses/51649/quizzes/158899

Review quizzes based on class material each day.

Homework assignment 1 due Thursday.