Recording will be on Canvas in My Media

REP(L) = $\{w \in T^* \mid$ between every pair of successive 2s in w is a string in $L\}$

No matter what L is, we can conclude that each in REP(L)

$L \subseteq \{0,1\}^*$

$REP(L) \subseteq \{0,1,2\}^*$

ε 0 2 do not have a pair of successive 2s.

Q: Is it true that for all sets X, $\emptyset \cup X = X$?

Note ε is not necessarily an element of sets. In particular $\varepsilon \notin \emptyset$. "The empty string is not an element of the empty set"

By definition $\emptyset \cup X = \{w \mid w \in \emptyset \text{ or } w \in X\} = \{w \mid w \in X\} = X$

($w \in \emptyset$: never true)

HW Q1e "Write a ~~template~~ for a regular expression that describes ..."

For example to write a template for a regular expression that describes the language that results from taking all strings in some language L over $\{0,1\}$ that is described by R and appending a 0 to the end of each one of them, we could use

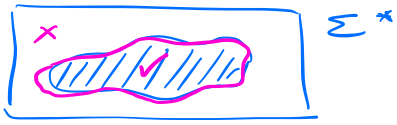
Template: R 0

Notice R^*0 or $R0^*$ don't work as template for this language.

Q: Does hw need to be in LaTeX?
A: No but it needs to be typed.

Textbook references: Within a chapter, each item is numbered consecutively. Figure 1.22 is the twenty-second numbered item in chapter one; it comes right after Example 1.21 and right before Definition 1.23.

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



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 describing patterns in strings. DFA 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 DFA that recognizes it. Another way of thinking about it: properties described by regular expressions require exactly the computational power of DFAs.

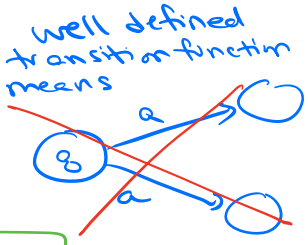
Monday

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

multiple representations

- Finite set of states Q
- Alphabet Σ
- Transition function δ
- Start state q_0
- Accept (final) states F

DFA well defined transition function means



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

$\delta((q, ?)) = ?$

$\delta((q, ?)) = ?$

$|\Sigma|$

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

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

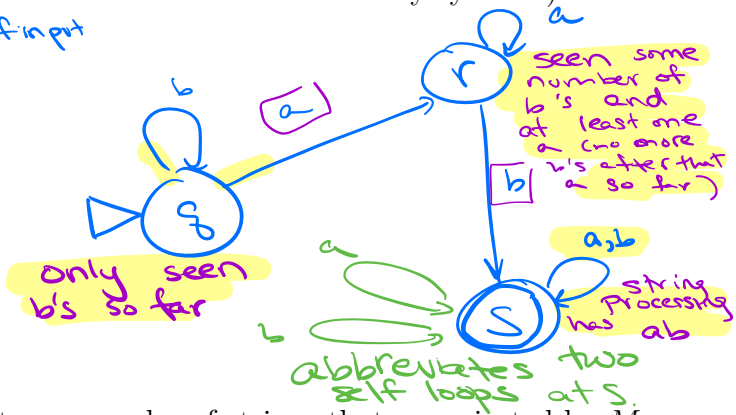
Σ exactly one state in the set of accept states

char of input

δ	a	b
q	r	q
r	r	s
s	s	s

current state

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:

bab abb ab

Add "labels" for states in the state diagram, e.g. "have not seen any of desired pattern yet" or "sink state".

We can use the analysis of the roles of the states in the state diagram to describe the language recognized by the DFA.

$L(M) =$ $\{w \in \{a,b\}^* \mid ab \text{ is a substring of } w\}$

shorthands ok to replace aa^* with a^+ and to replace $(a \cup b)^*$ with Σ^* when $\Sigma = \{a, b\}$

A regular expression describing $L(M)$ is

$b^* a a^* b (a \cup b)^*$

$(a \cup b)^*$ prefix $\quad a\ b$ substring $\quad (a \cup b)^*$ suffix.

Notice $bb^* aa^* (a \cup b)^*$ describes a different language because

$\tilde{L}$ = $L(bb^* aa^* (a \cup b)^*)$ (concatenation)

$ab \in L(M)$ but $ab \notin \tilde{L}$ All strings in $\tilde{L}$ start with b but ab doesn't.

justification: computation of M on ab is $q_0 q_1 q_2$, $q_2 \in F$ so M accepts ab.

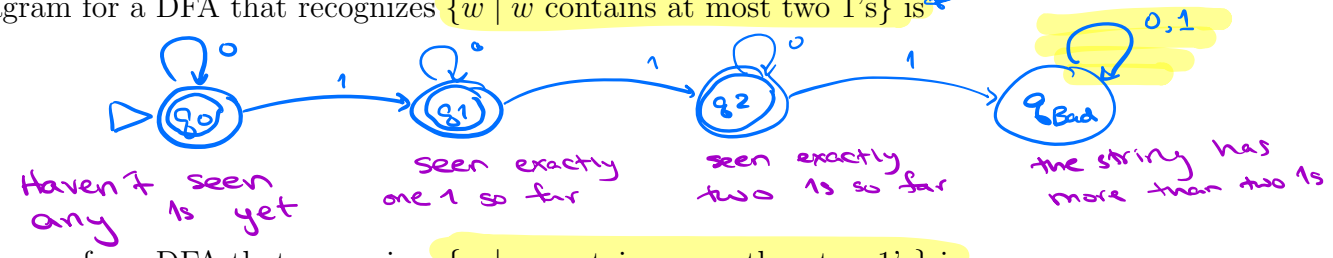
so ab is a counterexample to set equality.

a computation of a FA on a string is a sequence of states

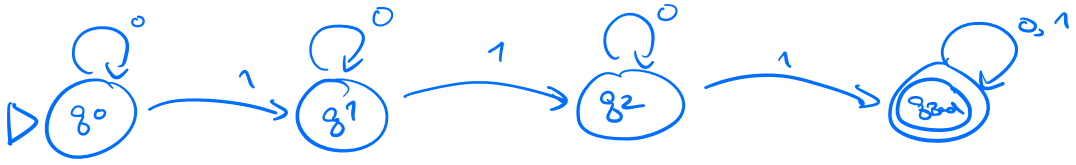
Let the alphabet be $\Sigma_1 = \{0, 1\}$.

Design Problem

A state diagram for a DFA that recognizes $\{w \mid w \text{ contains at most two 1's}\}$ is



A state diagram for a DFA that recognizes $\{w \mid w \text{ contains more than two 1's}\}$ is



Extra example: A state diagram for DFA recognizing

$$\{w \mid w \text{ is a string over } \{0,1\} \text{ whose length is not a multiple of } 3\}$$

Let n be an arbitrary positive integer. What is a formal definition for a DFA recognizing

$$\{w \mid w \text{ is a string over } \{0,1\} \text{ whose length is not a multiple of } n\}?$$

Review: Week 2 Monday

Please complete the review quiz questions on Gradescope about the languages recognized by DFAs.

Recall: Review quizzes based on class material are assigned each day. These quizzes will help you track and confirm your understanding of the concepts and examples we work in class. Quizzes can be submitted on Gradescope as many times (with no penalty) as you like until the quiz deadline: the three quizzes each week are all due on Friday (with no penalty late submission open until Sunday).

Pre class reading for next time: Pages 45-47.

Wednesday

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:

Proof:

A useful (optional) bit of terminology: the **iterated transition function** of a DFA $M = (Q, \Sigma, \delta, q_0, F)$ 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 $\delta^*(\ (q_0, w)\ ) \in F$.

Fix $\Sigma = \{a, b\}$. A state diagram for a DFA that recognizes $\{w \mid w \text{ has } ab \text{ as a substring and is of even length}\}$:

Suppose A_1, A_2 are languages over an alphabet Σ. **Claim:** if there is a DFA M_1 such that $L(M_1) = A_1$ and DFA M_2 such that $L(M_2) = A_2$, then there is another DFA, let's call it M, such that $L(M) = A_1 \cap A_2$.

Proof idea:

Formal construction:

Application: When $A_1 = \{w \mid w \text{ has } ab \text{ as a substring}\}$ and $A_2 = \{w \mid w \text{ is of even length}\}$.

Suppose A_1, A_2 are languages over an alphabet Σ. **Claim:** if there is a DFA M_1 such that $L(M_1) = A_1$ and DFA M_2 such that $L(M_2) = A_2$, then there is another DFA, let's call it M, such that $L(M) = A_1 \cup A_2$. *Sipser Theorem 1.25, page 45*

Proof idea:

Formal construction:

Application: A state diagram for a DFA that recognizes $\{w \mid w$ has ab as a substring or is of even length$\}$:

Review: Week 2 Wednesday

Please complete the review quiz questions on Gradescope about the languages recognized by DFAs.

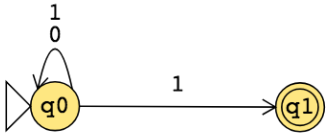
Recall: Review quizzes based on class material are assigned each day. These quizzes will help you track and confirm your understanding of the concepts and examples we work in class. Quizzes can be submitted on Gradescope as many times (with no penalty) as you like until the quiz deadline: the three quizzes each week are all due on Friday (with no penalty late submission open until Sunday).

Pre class reading for next time: Introduction to Section 1.2

Friday

Nondeterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$	
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	if and only if **there is** a computation of M on the input string that processes the whole string and ends in an accept state.
Page 53	

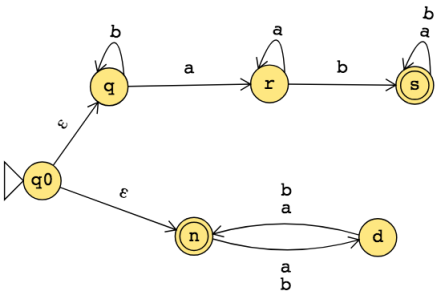
The formal definition of the NFA over $\{0, 1\}$ given by this state diagram is:



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

Change the transition function to get a different NFA which accepts the empty string.

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:

Review: Week 2 Friday

Please complete the review quiz questions on Gradescope about NFA.

Pre class reading for next time: Theorem 1.47, Theorem 1.49

