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 $\boxed{L}$ is, we can conclude that each in $\boxed{REP(L)}$

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

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

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 $\emptyset \cup X = \{ w \mid \underbrace{\boxed{w \in \emptyset}}_{\text{never true}} \text{ or } w \in X \} = \{ w \mid w \in X \} = X$

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

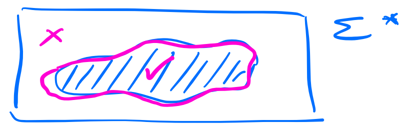
Template: $R\,0$

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

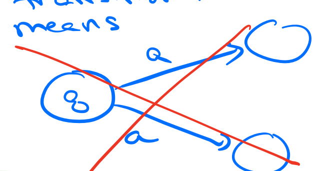
Output: Yes/ No, where Yes means that the input string matches the pattern or property described by the problem.

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

DFA well defined transition function means



- Finite set of states Q
- Alphabet Σ
- Transition function δ $\delta : Q \times \Sigma \to Q$ $\delta((q, ?)) = ?$
- Start state q_0
- Accept (final) states F $\delta((q, ?)) = ?$

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

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

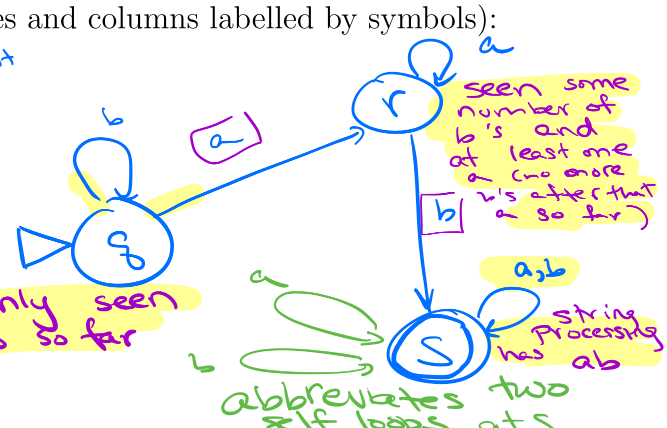
Σ exactly one state in the set of accept states

char of input

current state

δ	a	b
q	r	q
r	r	s
s	s	s

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

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

$$\underbrace{(a \cup b)^*}_{\text{prefix}} \underbrace{a \quad b}_{\text{substring}} \underbrace{(a \cup b)^*}_{\text{suffix.}}$$

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

justification

$ab \in L(\underline{M})$ $\boxed{\text{but}}$ $ab \notin \underbrace{L(\overbrace{bb^* aa^* (a \cup b)^*}^{\text{concatenation}})}_{\tilde{L}}$

computation of M on ab is q, r, s, $s \in F$ so M accepts ab. All strings in $\tilde{L}$ start w/ b but ab doesn't.

so ab is a counterexample to set equality.

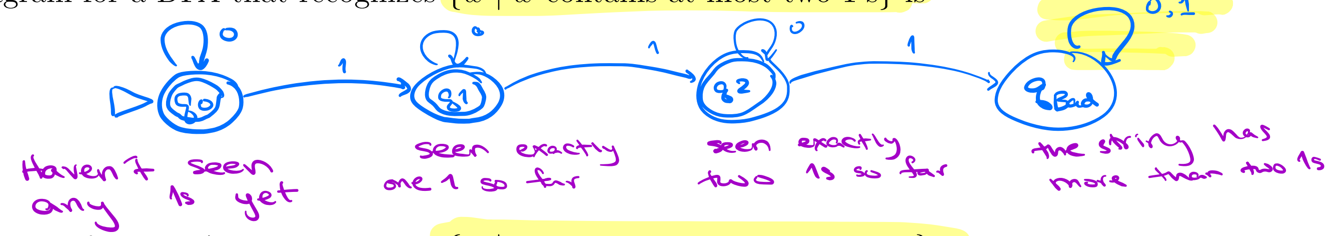
a computation of a FA on a string is a sequence of states

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

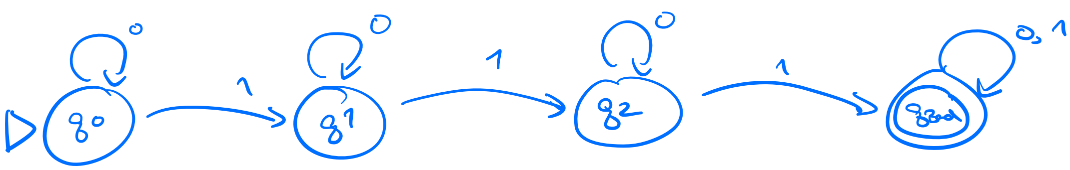
Design Problem

sink state

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



How are these languages related?

$\{w \mid w \text{ contains at most two 1s}\} = \overline{\{w \mid w \text{ contains more than two 1s.}\}}$

How are the DFAs related?

accept states and states that are not in set of accept states switch roles.

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

universal

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 $\{w \in \Sigma^* \mid w \notin A\}$. existential

Proof idea: Switch role of accept and not accept states from M to build M'.

Proof:

Let A be an arbitrary language over Σ.
Assume there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ for which $L(M) = A$.
We want to build M' so that $L(M') = \overline{A}$.
Define $M' = (\underbrace{Q}_{\text{same set of states}}, \underbrace{\Sigma}_{\text{same alphabet}}, \underbrace{\delta}_{\text{same arrows}}, \underbrace{q_0}_{\text{same start}}, \underbrace{\{q \in Q \mid q \notin F\}}_{\text{switch roles!}})$

Claim that $L(M') = \overline{A}$.

Goal ① WTS $L(M') \subseteq \overline{A}$. Let w be an arbitrary string accepted by M'. By definition, this means that the computation of M' on w ends in a state, let's call it r, and $r \in \underbrace{\{q \in Q \mid q \notin F\}}_{\text{set of accept states of } M'}$ i.e. $r \notin F$. Because M and M' share set of states, start state and transition function, computation of M on w also ends in r. Since $r \notin F$ M rejects w. By definition $L(M) = A$. So since $w \notin L(M)$, also $w \notin A$, i.e. by definition of complement $w \in \overline{A}$.

Goal ② WTS $\overline{A} \subseteq L(M')$. (keeps going next page)

Review 8/12

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

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

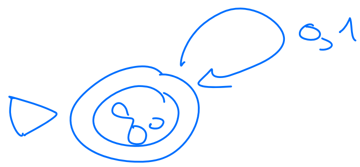
Using this terminology, M accepts a string w over Σ if and only if $\delta^*(\ (q_0, w)\ ) \in F$.

Let y be an arbitrary string in $\overline{A}$.
We want to show $y \in L(M')$, namely that M' accepts y. By assumption $A = L(M)$ so $y \in \overline{A}$ means $y \notin A$ and $y \notin L(M)$. In other words, M rejects y. Using the iterated transition function terminology, $\delta^*((q_0, y)) \notin F$.
By definition of M' (having same δ and q_0 as M, and flipped accept states), this mean $\delta^*((q_0, y))$ is an accept state in M' so M' accepts y, as required.

Since we proved $L(M') \subseteq \overline{A}$ and $L(M') \supseteq \overline{A}$, we have $L(M') = \overline{A}$, so M' is a DFA that recognizes $\overline{A}$, as required.

Notice: if $A = \Sigma^*$, $\Sigma = \{0,1\}$

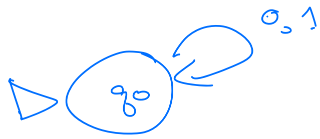
M



$M = (\{q_0\}, \{0,1\}, \delta, q_0, \{q_0\})$

$\delta(\_ \;\_\_\_\_)$

M'



$M' = (\{q_0\}, \{0,1\}, \delta, q_0, \emptyset)$

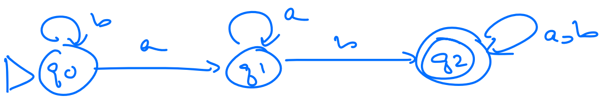
$\delta(\_ \;\_\_)$

$L(M') = \emptyset$

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

Notice this language is $\underbrace{\{w \mid w \text{ has } ab \text{ as a substring}\}}_{A_1} \cap \underbrace{\{w \mid w \text{ is of even length}\}}_{A_2}$

M_1 with $L(M_1) = A_1$



M_2 with $L(M_2) = A_2$.

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: Run computation of M_1 and M_2 in parallel

Data structure: ordered pairs

Formal construction:

Let A_1, A_2 be languages over Σ and suppose

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ is such that $L(M_1) = A_1$ and

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ is such that $L(M_2) = A_2$

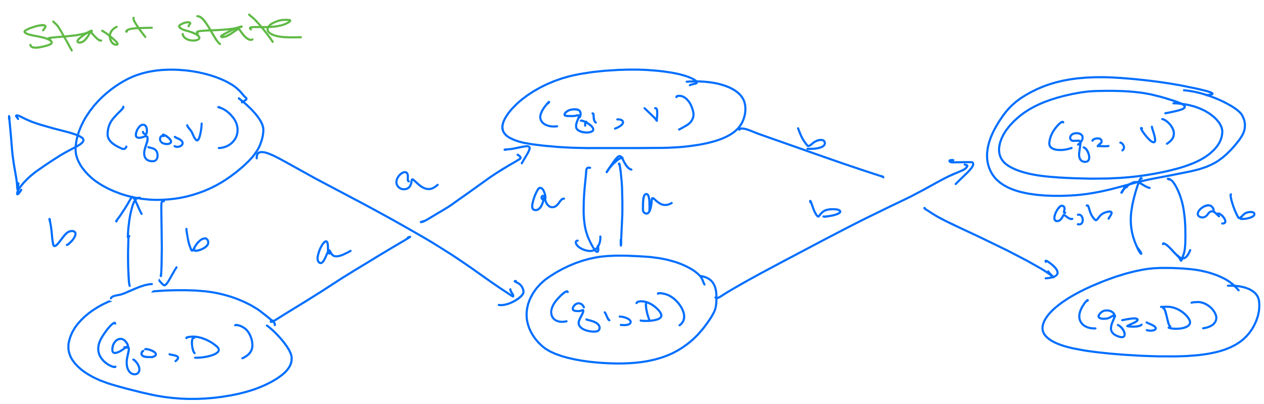
Define $M = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$ where $\delta : Q_1 \times Q_2 \times \Sigma \to Q_1 \times Q_2$

and $\delta(\underbrace{(x,y)}_{\text{state in } Q_1,\ \text{state in } Q_2}, \underbrace{\sigma}_{\text{input character}}) = (\underbrace{\delta_1(x,\sigma)}_{\text{next state in } Q_1}, \underbrace{\delta_2(y,\sigma)}_{\text{next state in } Q_2})$ for $(x,y) \in Q_1 \times Q_2$, $\sigma \in \Sigma$

σ is lower case sigma

Σ is upper case sigma

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



Proof of correctness of formal construction left as exercise.

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 $\{w \mid w \text{ has } ab \text{ as a substring or is of even length}\}$:

Review: Week 2 Wednesday

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

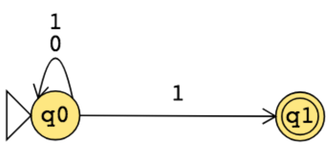
Recall: Review quizzes based on class material are assigned each day. These quizzes will help you track and confirm your understanding of the concepts and examples we work in class. Quizzes can be submitted on Gradescope as many times (with no penalty) as you like until the quiz deadline: the three quizzes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

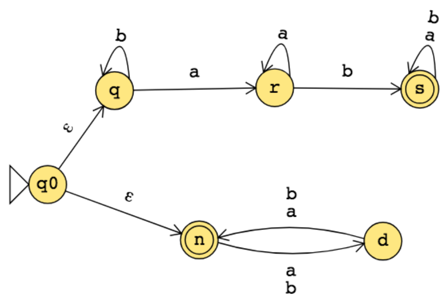
The formal definition of the NFA over $\{0, 1\}$ given by this state diagram is:



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

