

Recording will be on Canvas in My Media

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

No matter what L is, we can conclude that each in $REP(L)$ $L \subseteq \{0,1\}^*$ $REP(L) \subseteq \{0,1,2\}^*$

ϵ \uparrow 0 \uparrow 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 $\epsilon \notin \emptyset$. "The empty string is not an element of the empty set."

By definition $\emptyset \cup X = \{w \mid w \in \emptyset \text{ or } w \in X\} = \{w \mid w \in X\} = X$
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: $R0$

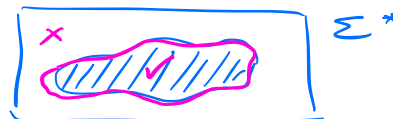
Notice R^*0 or $R0^*$ don't work as template for this language.

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: $M = (Q, \Sigma, \delta, q_0, F)$

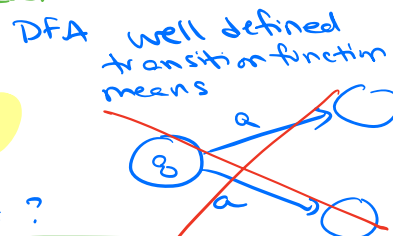
multiple representations

- Finite set of states Q
- Alphabet Σ
- Transition function δ

$$\delta: Q \times \Sigma \rightarrow Q$$

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

- Start state q_0
- Accept (final) states F



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

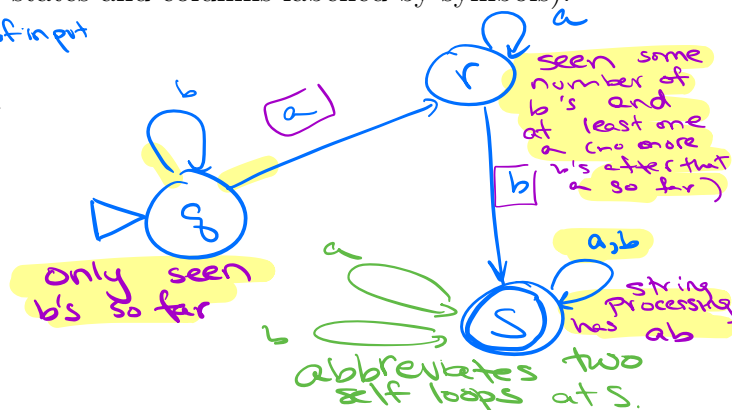
$$|\Sigma|$$

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

Σ ↑ exactly one state in the set of accept states

δ	char of input	
	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.

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

A regular expression describing $L(M)$ is

$$\underline{b^* a a^* b (a \cup b)^*}$$

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

shorthands ok to replace

aa^*

with

a^+

and to replace

$(a \cup b)^*$ with Σ^* when $\Sigma = \{a, b\}$

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

justification

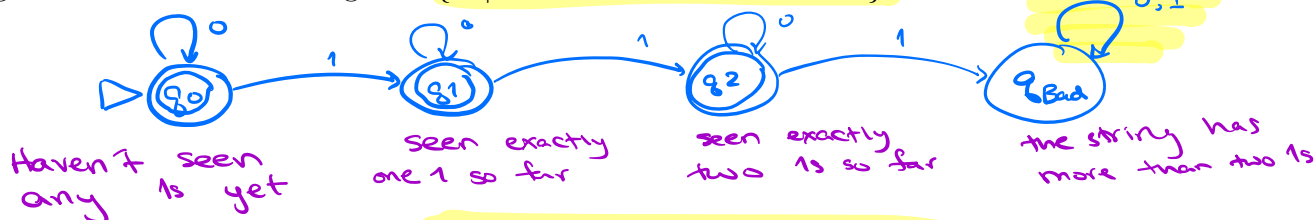
computation of M on ab is q_0, q_1, q_2, q_0 so ab is a counterexample to set equality.

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

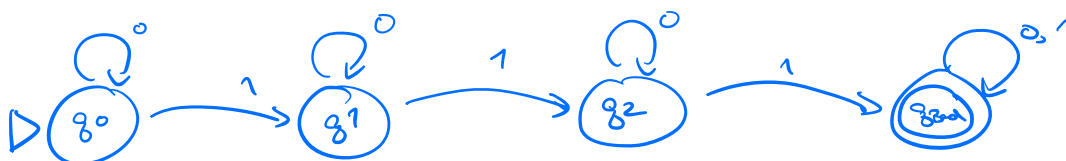
Let the alphabet be $\Sigma_1 = \{0, 1\}$.

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



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 $L(M') = \bar{A}$, 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 $L(M') = \bar{A}$.

Define $M' = (Q, \Sigma, \delta, q_0, \{q \in Q \mid q \notin F\})$
Same Set of states same alphabet same arrows same start Switch roles!

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

Goal ① WTS $L(M') \subseteq \bar{A}$. 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 $r \in \{q \in Q \mid q \notin F\}$ 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 $w \in \bar{A}$.

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

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

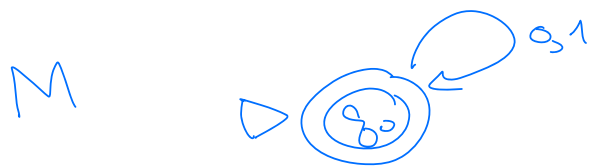
$$\delta^*((q, w)) = \begin{cases} q & \text{if } q \in Q, w = \varepsilon \\ \delta((q, a)) & \text{if } q \in Q, w = a \in \Sigma \\ \delta((\delta^*(q, u), a)) & \text{if } q \in Q, w = ua \text{ where } u \in \Sigma^* \text{ and } a \in \Sigma \end{cases}$$

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 \bar{A} .
 We want to show $y \in L(M')$, namely
 that M' accepts y . By assumption
 $A = L(M)$ so $y \in \bar{A}$ 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 means $\delta^*(q_0, y)$ is
 an accept state in M' so M' accepts y ,
 as required.

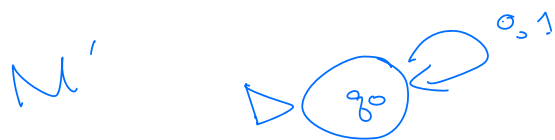
Since we proved $L(M') \subseteq \bar{A}$ and
 $L(M') \supseteq A$, we have $L(M') = \bar{A}$, so
 M' is a DFA that recognizes \bar{A} ,
 as required.

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



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

$$\delta(_ _)$$



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

$$\delta(_ _)$$

$$L(M') = \emptyset$$

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 $L(M_2) = A_2$

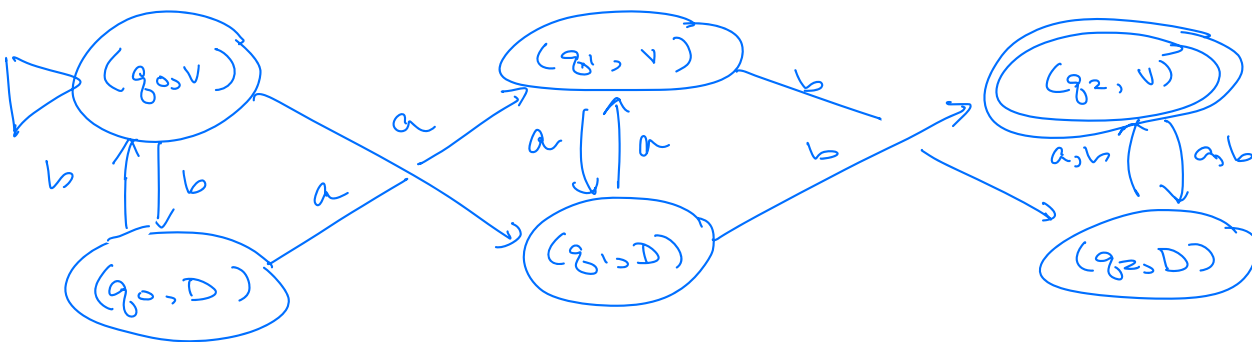
Define $M = (Q, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$ where $\delta: Q \times \Sigma \rightarrow Q$

and $\delta((x, y), \sigma) = (\delta_1(x, \sigma), \delta_2(y, \sigma))$ 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 $A_2 = \{w \mid w \text{ is of even length}\}$.

Start state



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

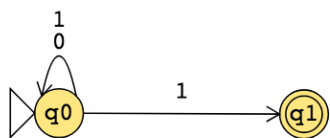
Friday

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: q_0, q_1, \dots
Alphabet Σ	Each input to the automaton is a string over Σ .
Arrow labels Σ_ϵ	$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$.
Transition function δ	Arrows in the state diagram are labelled either by symbols from Σ or by ϵ . $\delta : Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q)$ 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:

$\epsilon \neq \emptyset$
 $\{\} = \emptyset$



Why is this not state diagram of DFA?
 There are two outgoing arrows from q_0 (labelled 0 and 1)
 There is no outgoing arrow from q_1 (labelled 0)
 There is no outgoing arrow from q_1 (labelled 1)

$(\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$

	δ	arrow labels		
		0	1	ϵ
states	q_0	$\{q_0\}$	$\{q_0, q_1\}$	\emptyset
	q_1	$\{\}$	\emptyset	\emptyset

Example

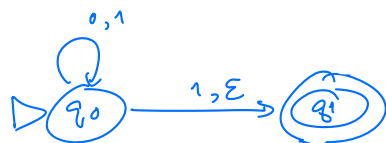
The language over $\{0, 1\}$ recognized by this NFA is:

The set of strings that end in 1

input string 011
 Possible computation: $q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0$ ENDS NOT IN F
 Possible computation: $q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_1$ ENDS IN F
 Possible computation: $q_0 \rightarrow q_0 \rightarrow q_1$ STUCK!

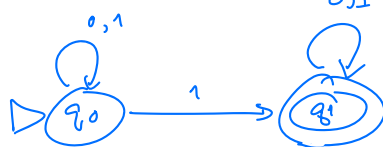
$$\{w \in \{0, 1\}^* \mid w \text{ ends in } 1\} = L(\Sigma^* 1)$$

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



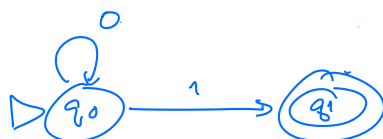
Note: - this changes the language in a big way. This NFA accepts ϵ , and all other strings.

Notice this other NFA



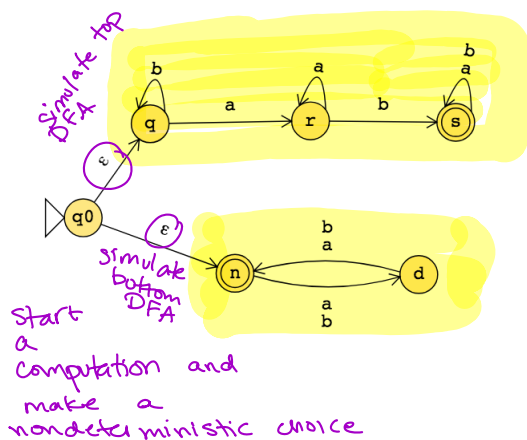
accepts 010

Notice this other NFA



accepts 01, rejects 011

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



Accepts
 ab
 ba
 ε
 aba
Rejects.
 bbb
 a

$(\{q_0, q, r, s, n, d\}, \{a, b\}, \delta, q_0, \{s, n\})$

where δ is given by

	a	b	ε
q ₀	∅	∅	{q, n}
q	{r}	{q}	∅
r	{r}	{s}	∅
s	{s}	{s}	∅
n	{d}	{n}	∅
d	{n}	∅	∅

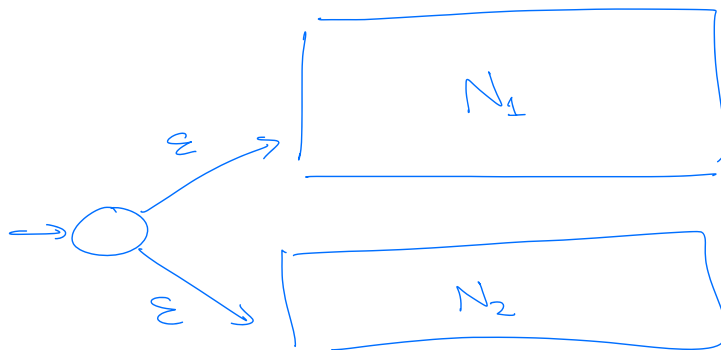
The language recognized by this NFA is:

$\{w \in \{a, b\}^* \mid w \text{ has } ab \text{ as a substring} \text{ OR } w \text{ has even length}\}$

Thm: Whenever L_1, L_2 are languages recognized by NFA N_1, N_2 , there is a NFA N with $L(N) = L_1 \cup L_2$.

Pf idea: For L_1, L_2 languages over Σ and given NFA N_1, N_2 with $L_1 = L(N_1)$ $L_2 = L(N_2)$

Define new NFA N according to diagram



and prove $L(N) = L(N_1) \cup L(N_2)$.

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