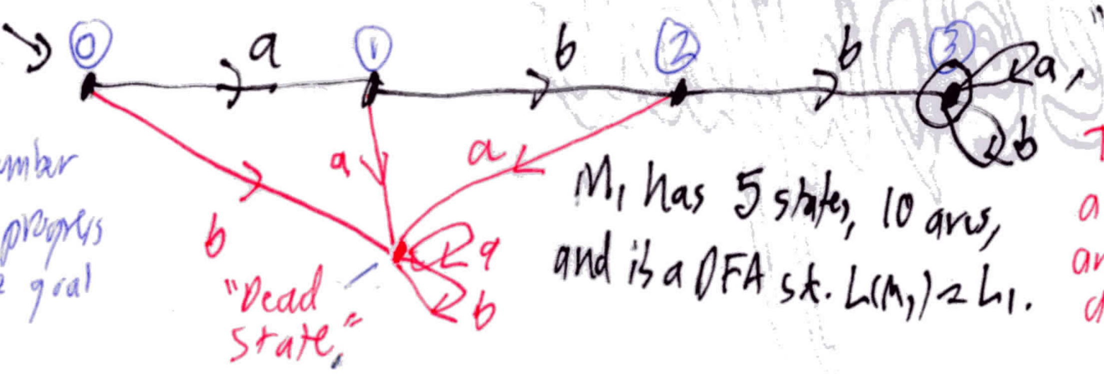
CSE396 **Lecture Tue Feb 13** Spring 2018

0697 $L_1 = \{x \in \Sigma^* : x \text{ begins with } abb\}$ Design DFAs M_1 and M_2 s.t.

Let $\Sigma = \{a,b\}$. $L_2 = \{x \in \Sigma^* : x \text{ ends with } abb\}$. $L(M_1) = L_1$ & $L(M_2) = L_2$.

M_1:



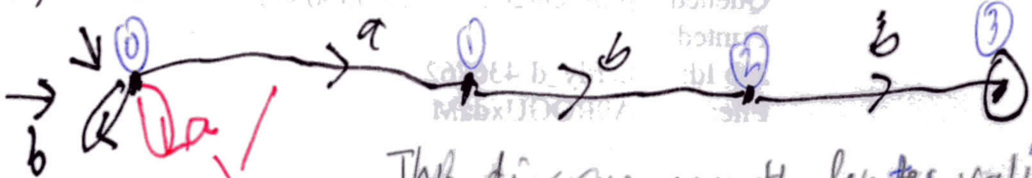
state number indicates progress toward the goal

M_1 has 5 states, 10 arcs, and is a DFA s.t. $L(M_1) = L_1$.

To "complete" this to a DFA, we need an arc for every state and character.

Mirror Image Strategy for M_2

NFA N_2



$\delta_2 = \{$
$(0,a,0)$
$(0,a,1)$
$(0,b,0)$
$(1,b,2)$
$(2,b,3)\}$

Nondeterminism on a at the start state

Example: $x = aabbabb$

This diagram correctly denotes valid computation paths:

$\vec{c} = (0, a, 0, a, 1, b, 2, b, ③$ next char is a. We cannot process it, so this cannot be completed to a valid computation on x.

Try again: $\vec{c}\,' = (0, a, 0, a, \underline{0}, b, 0, b, 0, a, 1, b, 2, b, ③)$ OK since we processed all of x.

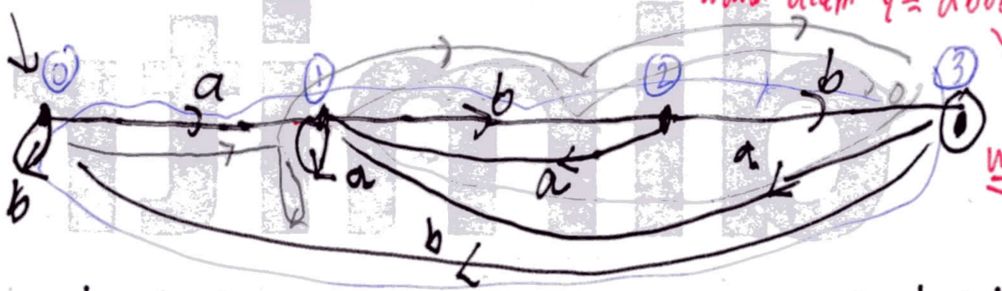
Thus $\vec{c}$ is a valid accepting computation of M_2 on x, so $x \in L_2$. And $\boxed{L(M_2) = L_2.}$

Defn: An NFA (without ε-arcs) has the same "class object defn" as a DFA, with $\delta \subseteq Q \times \Sigma \times Q$ without requiring δ to be a function from $Q \times \Sigma$ to Q

Build a DFA M_2 for L_2:

Consider also $x' = aabbbabb$

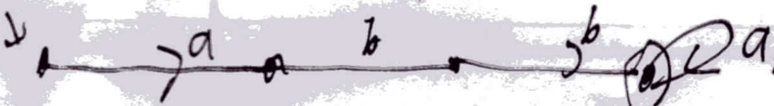
If we had $\delta_2(0,b) = ①$, then the resulting DFA M_2' would accept $y = abbbbb$. But $y \notin L_2$, so M_2' is unsound.



This is a legal DFA. It is correct: $L(M_2) = L_2$

It has no dead state and no "nirvana" state.

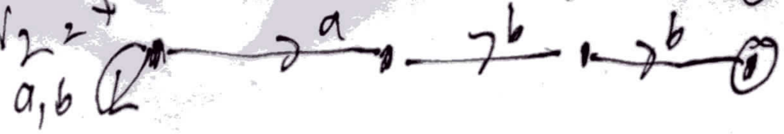
Our NFAs: $N_1 =$



N₁ counts as an NFA since it lacks the arcs to a dead state, but it has no nondeterminism

They suggest shorthand for L_1 and L_2.

$N_2 =$



$$L_1 = a\cdot b\cdot b\cdot \boxed{(a \cup b)^*}$$

Begins with a b b followed by zero or more occurrences of chars which can each be a or b

$$L_2 = (a\cup b)^* \cdot a\cdot b\cdot b.$$

How about $R_3 = (a\cup b)^* abb (a\cup b)^*$

What language does this regular expression denote? $L(R_3) = \{x \in \{a,b\}^* : x \text{ has } abb \text{ as a substring}\}$.

$L(R_3)$ includes $L_1 \cup L_2$ and other strings like $z = aabba$.

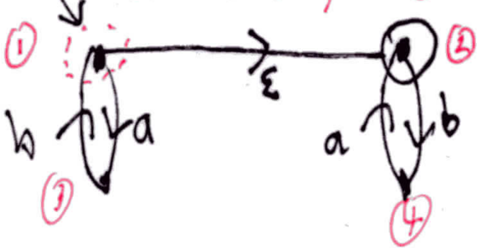
Now consider $R_4 = (a\cdot b)^* \cdot (ba)^*$. What language does R_4 denote?

is $z = aabba \in L(R_4)$? No. already dead here

We can diagram a corresponding NFA, using an ε-arc

OK not to count ① as an accepting state, since it can "freely" go to ②.

N_4



$w = abbabb$? OK bad: We needed another a.

How about ε? Yes: both $*$s give "zero".

abab, bababa, similarly belong.

Def^n: An NFA (with ε-arcs) is a 5-tuple $N = (Q, \Sigma, \delta, s, F)$ with $s \in Q$ and $F \subseteq Q$ as before, but now $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$

N_4 has $s = 1$, $F = \{2\}$, and [Text: $\delta : (Q \times \Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$.]

$\delta = \{(1,a,3), (3,b,1), (1,\varepsilon,2), (2,b,4), (4,a,2)\}$. Computations similarly liberalize:

$\vec{c} = (1, a, 3, b, 1, \varepsilon, 2, b, 4, a, 2)$ is a valid computation that processes $a\cdot b\cdot \varepsilon \cdot b \cdot a = abba$ from state 1 to state 2. $L(N_4) = L(R_4)$ ~~can't ... x ... ...~~

END