Announcements:

- Virtual lectures today & Thursday
  (Virtual O hours on Thursday)
- HW 1 due today @ 11:59pm
- HW 2 due next Tuesday (up soon)

Minor grading / late updates:

- Still 3 late days.
  (If you need more — email me by Tues)
- grade floor 50%.
- similarly: completely missed assignments get a score = average over all other assignments — 25%.
  @ end of term

  (90, 90, 0, 90, 90) → 72%

  (90, 90, 65*, 90, 90) → 85%

- ODS note-taker.

Today:

0. Review: DFAs, NFAs, regular languages & operations.
1. Reduce NFAs to DFAs. (not necessary for HW Q3).
   ↳ NFAs, DFAs both see regular languages.
2. Proofs that regular languages are closed under ∘, ∪, * using NFAs.

3. Regular expressions.

Rapid review:

Language := a set of strings (over some alphabet)

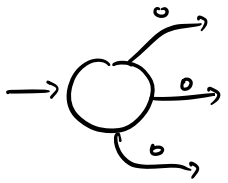
Automata := math machines that take input strings, compute, and answer yes/no.

→ **DFAs:**

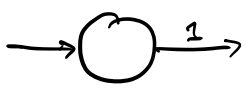
- one branch of computation
- exactly one transition from each state for each alphabet symbol.
- accepted if we end at ◎ when computation finishes
- accept regular languages.

→ **NFAs:**

- multiple branches of computation.
- ≥ 0 branches from each state on each alphabet symbol.

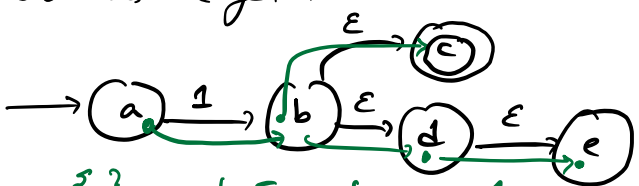


on 0: split into 3 branches.



on 0: branch "dies" (transition to ∅).

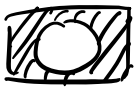
- ε-transitions! "free transitions / free splits", evaluated as soon as we get to them.

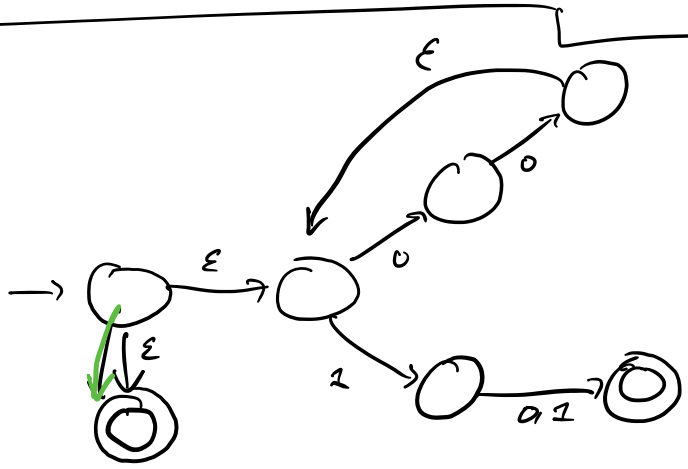


If in {a}, and I read in a 1:
first go to {b}, and then end up occupying {b, c, d, e} after ε-transitions.

- Accept if at least one live branch in ◎ when computation

end.

- Regular operations := if the input language(s) are regular, the output language is regular.
  - If A regular, $\overline{A}$ is regular  ✓
  - If A, B regular, $A \cup B$ regular  ✓
  - ——" ——, $A \cap B$ regular  ✓
  - If A, B regular, $A \circ B := \{wx \mid w \in A, x \in B\}$ is regular (to do).
  - If A regular, $A^* := \{a_1 a_2 \cdots a_k \mid \text{all } a_i\text{s} \in A, k \geq 0\}$ is regular (to do).



101

1. Turning NFAs into DFAs.

(Prop 1. We can convert DFAs to NFAs.
Proof. DFA state diagrams are NFA state diagrams.)

Prop 2. We can convert any NFA into an equivalent DFA.

Idea: Take an NFA, simulate it w/ a DFA.

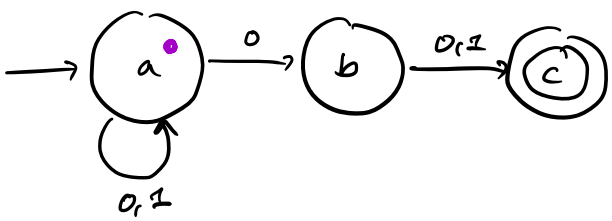
NFAs occupy a set of states at every time step

sets of states in NFA $\Longleftrightarrow$ states in DFA.

(s states) $\longleftrightarrow$ 2^s

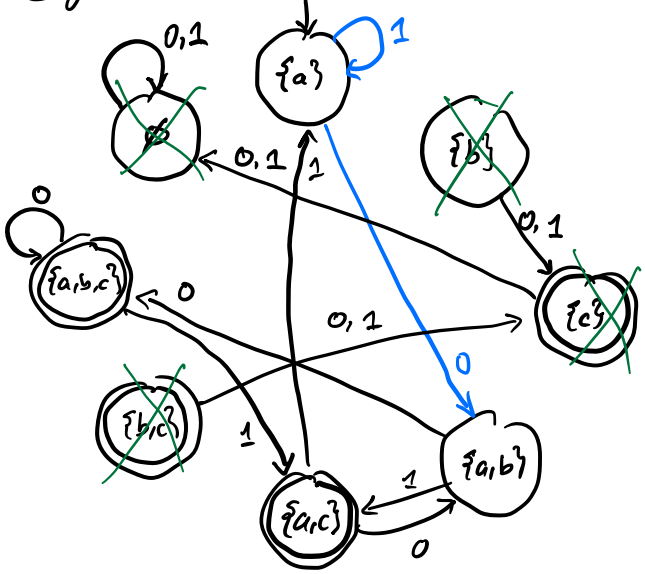
state set Q $\longleftrightarrow$ $\mathcal{P}(Q)$

Example:



$N = (Q = \{a,b,c\},$
$\Sigma = \{0,1\},$
$q_0 = a,$
$F = \{c\},$

Equivalent DFA D:



δ:

	0	1	ε
a	{a,b}	{a}	∅
b	{c}	{c}	∅
c	∅	∅	∅

* special case: states after {x} = $E(x)$.

$D = (Q' = \mathcal{P}(Q),$
$\Sigma = \{0,1\}$
$q_0 = \{a\}$
$F = \{\{c\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}$
δ: see picture)

Back in 5 - resume at 2:15.

If the NFA has ε-edges?

Above: transition from state $\{a,b\}$ to $\{a,b,c\}$ on a 0 if we end up in $\{abc\}$ after taking 0 edges from $\{a,b\}$ (and evaluating all ε-transitions).

Proof. (NFAs have equivalent DFAs.)

Let $N = (Q, \Sigma, q_0, F, \delta)$ be an NFA.

We'll build a DFA $D = (Q', \Sigma, q_0', F', \delta')$ that recognizes the same language.

$Q' = \mathcal{P}(Q)$,

Σ left unchanged,

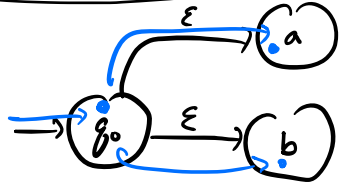
$F' = \{ R \in Q', \ R$ contains an accepting state in $N \}$.

$q_0' = E(\{q_0\})$

Finally, δ':

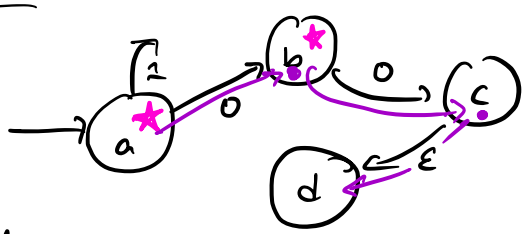
$$\delta'(R, a) = \{ q \in Q' \mid q \in E(\delta(r, a)) \text{ for } r \in R \}$$

(R is a set of NFA states; $a \in \Sigma$)



$E(\{q_0\}) = \{q_0, a, b\}$.

If S is a set of NFA states, then $E(S)$ (the "ε-closure") is the set of all states reachable from S by ε-transitions.



$\delta'(\{a,b\}, 0) = \{b, c, d\}$.

Argue: D accepts exactly the same set of strings as N.

Consider the execution of N, D on the string

$$w = w_1 w_2 \cdots w_k, \quad w_i \in \Sigma.$$

Show: after every step, N is in the set of states that matches the state of D.

Base case: N starts in $E(q_0)$,
D starts in $E(q_0)$

Inductive step: Suppose N occupies a set of states S after reading in character w_i and evaluating ϵ-transitions, and so does D.

Then, w_{i+1} : N occupy $E\left(\bigcup_{s \in S} \delta(s, w_{i+1})\right)$ (everywhere I can get to from S),

as does D (by definition.)

$$\{q \in Q' \text{ s.t. } q \in E(\delta(s, w_{i+1})) \text{ for any } s \in S\}$$

$\therefore$ at the end of execution, D occupies the state that matches the state set occupied by N, and accepts if that state set contains at least one (F). ∎

$$S \quad \bigcup_{s \in S} \delta(s, w_{i+1}) = \{q \text{ s.t. } q \in \delta(s, w_{i+1}) \text{ for any } s \in S\}$$

Takeaway:

= Languages recognized by NFAs

= Languages recognized by DFAs

= regular languages.

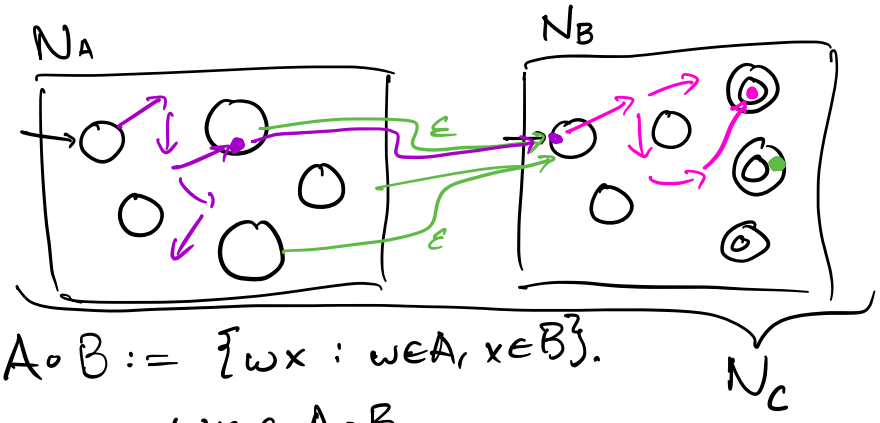
So: If NFA recognizes L, L is regular.

Helpful for proving closure under regular operations!

Theorem: If A regular, B regular, $A \circ B$ is regular.

Proof: Let $A = L(N_A)$, $B = L(N_B)$, where N_A, N_B are NFAs.

We'll build N_C, an NFA that recognizes $A \circ B$.



$A \circ B := \{wx : w \in A, x \in B\}$.

$wx \in A \circ B$.

- Add ε-transitions from accept states of N_A to start state of N_B
- Turn the accept states of N_A into regular states.

Claim: If $x \in A$, $y \in B$, then $xy = x \circ y \in L(N_C)$

If $x \in A$, then some branch of computation in N_A reaches an accept state after reading x and ε-transitions to the start of N_B.

Because $y \in B$, the computation ends w/ at least one branch in an accept state (of N_B, now N_C).

Claim: If N_C accepts w, then $w \in A \circ B$.

Logic: some branch of computation must reach an accept state of N_A, then take an ε-transition, then reach an accept state of N_B at the end of evaluation because $w \in L(N_C)$.

This means transitions taken by this branch in N_A $\circ$ ε $\circ$ transitions taken by this branch in N_B =

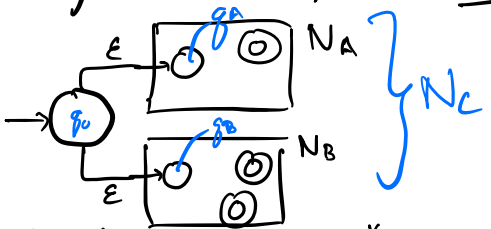
$$xy \mid x \in A, y \in B$$

Hints: 1) Start w/ 

Use powers of NFAs.

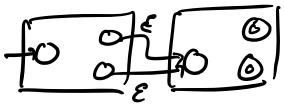
Puzzles: Show:

(1) If A, B regular, $A \cup B$ regular.

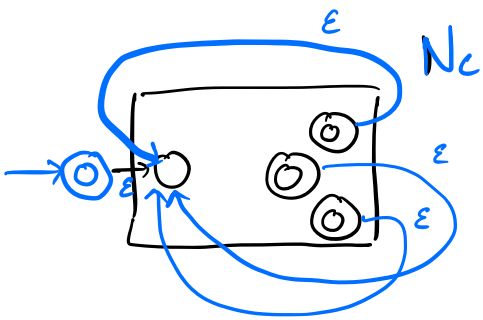


- connect new start state w/ ε-edges to q_A, q_B.
- on start: we're in $E(q_0) = \{q_0, q_A, q_B\}$

(Concatenation, for reference:)



(2) If A regular, A^* regular.



- connect accept states to start states w/ ε-transitions.
- say I have an input $a_1 a_2 a_3 \in A \circ A \circ A \in A^*$
- $\varepsilon \in A^*$: add a new start/accept state connected to the old start state w/ the ε-transition.

Proof sketches:

$A \circ B$
$A \cup B$
A^*
} all regular!

Takeaway: any language built from regular languages using the regular operations $(\overline{\ }, \cup, \cap, \circ, *)$ is regular!

$$(A \cap B) \cup (C^* \circ B)$$

If A, B, C regular ← all regular.

↳ $A \cap B$ regular

↳ C^* regular

↳ $C^* \circ B$ regular

Break: 3:10 - 3:15.

---

3. Regular Expressions.

regular expressions are another succinct way of describing a set of strings (a language).

Example: $(0 \cup 1)0^*$ // this is a special representation of a language — so symbols are used slightly differently.

read this as

"either a zero or a one, followed by some number of zeroes."

Example: $\{0,1\}^*$ (this is just the * operation on a language)

= the regular expression $(0 \cup 1)^*$.

Def. (Regular expression) R is a regular expression if it is any one of the following:

- $R = a$, for some $a \in \Sigma$. (Evaluates to $\{a\}$.)
- $R = \varepsilon$. (Evaluates to $\{\varepsilon\}$).

- $R = \emptyset$, (Evaluates to $\emptyset = \{\}$)
- $R = R_1 \cup R_2$, where R_1 and R_2 are shorter regular expressions. (example: $(0 \cup 1)$ ↓ ↓ $\{0\} \cup \{1\}$ ↓ $\{0, 1\}$)
- $R = R_1 \circ R_2$, where R_1, R_2 are regular expressions. $= R_1 R_2$ for brevity.
- $R = R_1^*$, where R_1 is a regular expression.

(example: $\underline{01^* \cup 1}$)

$0 \circ 1^*$, 1

- Write Σ as shorthand for "any symbol in the set/alphabet Σ"
  - $\Sigma = \{0, 1, 2\}$ (alphabet) regular expression $\Sigma 1$ ↓ $(0 \cup 1 \cup 2)1$
- Write R^+ as shorthand for RR^*, a.k.a. "at least one concatenated string from R"
- Write R^k for some finite $k \geq 0$ to mean "exactly k strings from R concatenated together."

$$R^k = \underbrace{RRR \cdots R}_{k \text{ times.}}$$

$(0 \cup 1)^{10}$ = all binary strings of length 10.

Order of operations:

(1) star, (2) concatenation, (3) union.

$$1 \cup 01^* = (1) \cup (0 \circ (1^*)).$$

Examples:

$$0^*10^* = \{w \in \{0,1\}^* \mid w \text{ contains a single } 1\}.$$

$$1^*(01^+)^* = \{w \in \{0,1\}^* \mid \text{every } 0 \text{ is followed by at least}$$

↳ "a zero followed by at least one 1" one 1 }.

$1^+ = 11^*$

$\underbrace{11\cdots1}_{1^*}\underbrace{01\cdots011\cdots01}_{01^+\ 01^+\ 01^+}$

$(\Sigma\Sigma\Sigma)^*$, where $\Sigma = \{0,1\}$.

Σ is shorthand for $(0 \cup 1)$

$((0\cup1)(0\cup1)(0\cup1))^* = \{ w \in \{0,1\}^* \mid |w| \text{ is divisible by } 3 \}$.

ε, 011, 101, 101011, 111110

101011

$(0 \cup \varepsilon)(1 \cup \varepsilon) = \{0\varepsilon, 01, \varepsilon 1, \varepsilon\varepsilon\}$

$= \{0, 01, 1, \varepsilon\}$

$\{w \mid w$ starts and ends with the same symbol$\}$?

$(0\Sigma^*0) \cup (1\Sigma^*1) \cup 0 \cup 1$.

↓ shortest match: 0ε0, 00

$D = \{0,1,2,\ldots 9\}$

$\Sigma = \{-, .\} \cup D$

regular expression for decimal numbers like 54, 9.5, −0.003, etc.

$(\varepsilon \cup -)(D^+ \cup (D^+.D^+))$

6.42

0.999

"special" cases: (1) anything concatenated with $\emptyset = \emptyset$

$1\emptyset$

↳ evaluate to

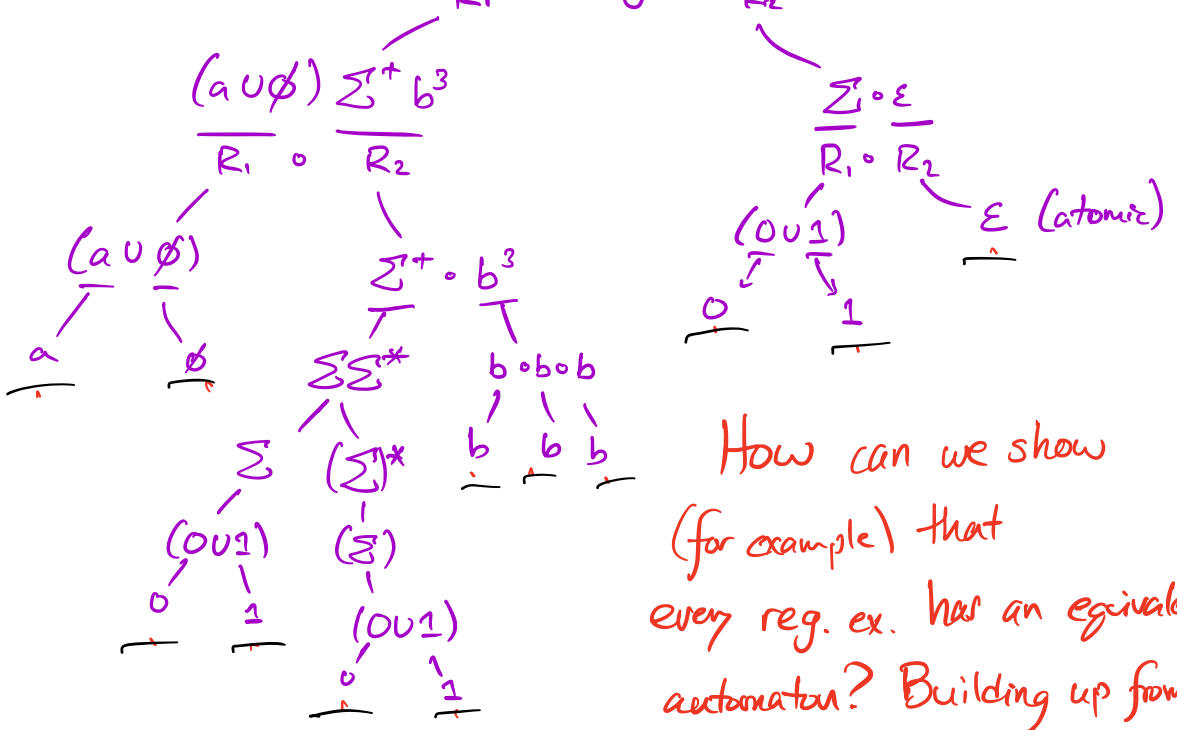
"Something from the set $\{1\}$" $\circ$ "something from the set $\{\}$"

(2) $\emptyset^* = \{\varepsilon\}$

"any number $k \geq 0$ things from $\{\}$ concatenated together"

Example: breaking down/reading a big reg ex.

$$R = \underbrace{((a \cup \emptyset)\Sigma^+ b^3)}_{R_1} \cup \underbrace{(\Sigma \circ \varepsilon)}_{R_2}, \quad \Sigma = \{0,1\}$$



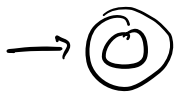
How can we show (for example) that every reg. ex. has an equivalent automaton? Building up from atoms.

Theorem (sketch) - we can build an NFA that accepts the same language as any regular expression R.

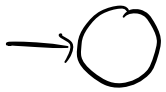
Proof. By (inductive) definition, every regular expression matches one of six cases.

(1) $a \in \Sigma$. (eval: $\{a\}$).

(2) ε. (eval: $\{\varepsilon\}$)

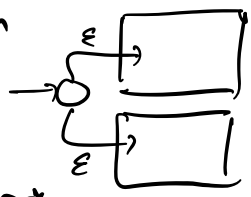


(3) $\emptyset$ (eval: $\emptyset$)



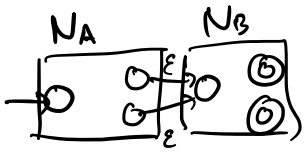
(4) $R_1 \cup R_2$.

If R_1 and R_2 evaluate to the languages $L(N_A)$ and $L(N_B)$, then

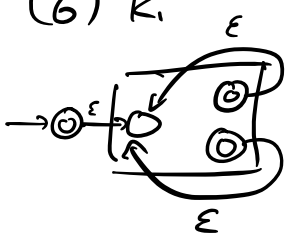


$= R_1 \cup R_2$

(5) $R_1 \circ R_2$



(6) R_1^*

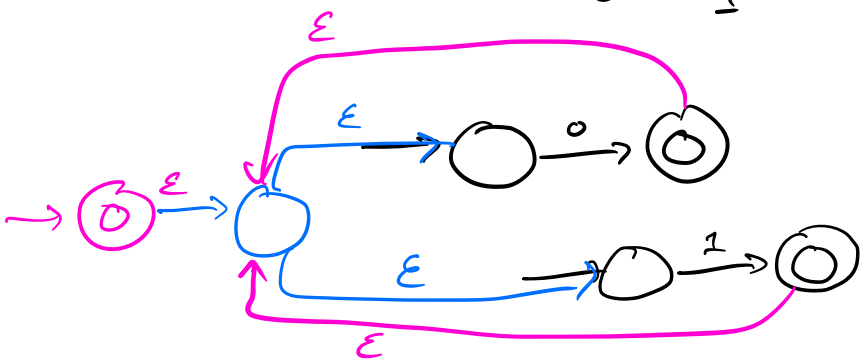


Example: $(0 \cup 1)^*$

R_1^*, $R_1 = (0 \cup 1)$

T T

0 1



More examples/regexes next time.

Reminders: HW 1 due tonight
HW 2 next Tues (cop Scan)

Two videos for this lecture:
- Reg. expressions
- Converting an NFA to a DFA*

$R = \varepsilon$, evaluates as $\{\varepsilon\}$ ↳ $\{""\}$

$R = \emptyset$, evaluate $\{\}$

Anything concat w/ $\emptyset = \emptyset$.

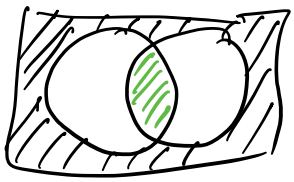
$01\emptyset \Longrightarrow$ evaluates $\{0\} \circ \{1\} \circ \{\} = \emptyset$

nothing to pick here.

$\{0\} \circ \{1\} \circ \{a, b\}$
$= \{01a, 01b\}$

$\{0\} \circ \{1\} \circ \{\varepsilon\}$
$= \{01\varepsilon\} = \{01\}$

HW 1 problem 4: Reg. Languages closed under XNOR.



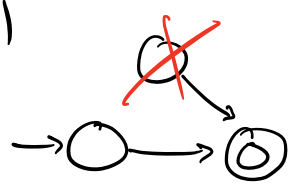
$/// = \overline{A} \cap \overline{B}$) ← new DFA "D_3" should accept if D_A, D_B both reject.

$/// = A \cap B$.

- If D_A accepts w ($w \in L(D_A)$) then we end in ◎.
- If D_A doesn't accept w ($w \notin L(D_A)$) ($w \in \overline{A}$), then we end ○.

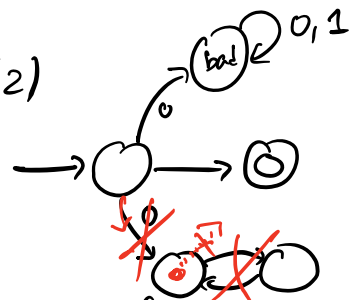
"Condensing" a NFA.

(1)



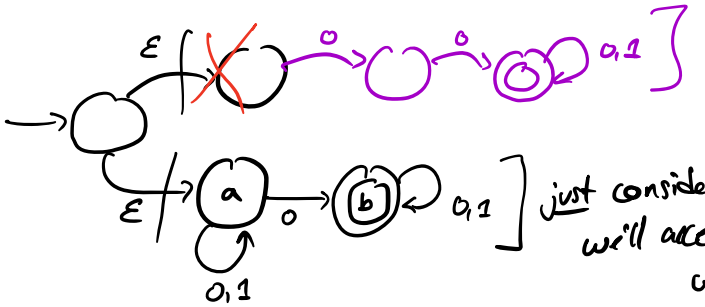
unreachable states.

(2)



states from which there is no accepting path.

(3) testing strings to find redundancies.

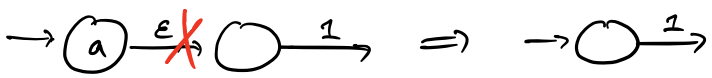


this branch accepts all strings that start with '00'.

just considering this branch, we'll accept on all strings with at least one zero.

How to realize this? Test strings to gain intuition.

(4)



How to check if simplification works? Test strings.