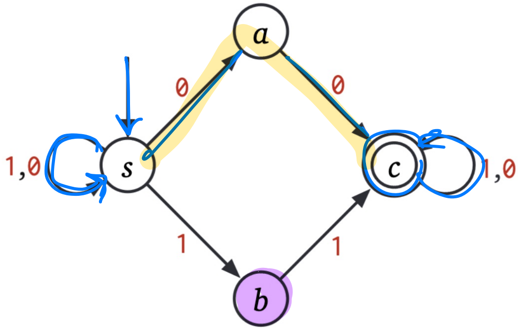
Goal 1: Prove that

language is regular <=> language is automatic

can write a regular expression — can give a DFA

regexp $\rightleftarrows$ DFA

Goal 2: Language Transformations



what's wrong w/ this DFA?

Non Deterministic Finite Automaton

NFA

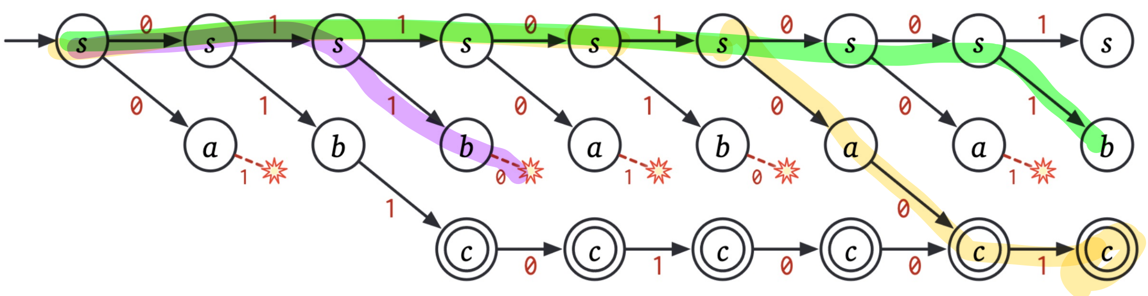
0 1101001

$s \xrightarrow{0} s \xrightarrow{1} s \xrightarrow{1} s \xrightarrow{0} s \xrightarrow{1} s \xrightarrow{0} a \xrightarrow{0} c \xrightarrow{1} c$

An NFA accepts string w if there is some accepting path for w.

Ways we can conceptualize an NFA:

- clairvoyance
- parallel threads



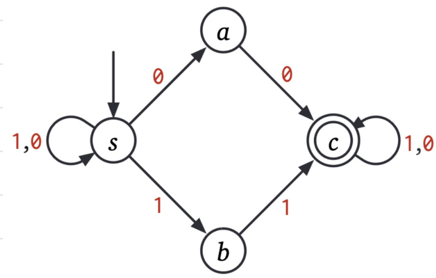
Running our example NFA on the input string 01101001.

- verification

NFA formal components:

Q : set of states

$\{s, a, b, c\}$



$s \in Q$

s

$\delta(s, 0) = \{s, a\}$

$A \subseteq Q$

$\{c\}$

power set of Q
= set of all subsets of Q

$\delta : Q \times \Sigma \to \mathcal{P}(Q)$ $\quad |\mathcal{P}(Q)| = 2^{|Q|}$

$2^4 = 16$

w/ partners:

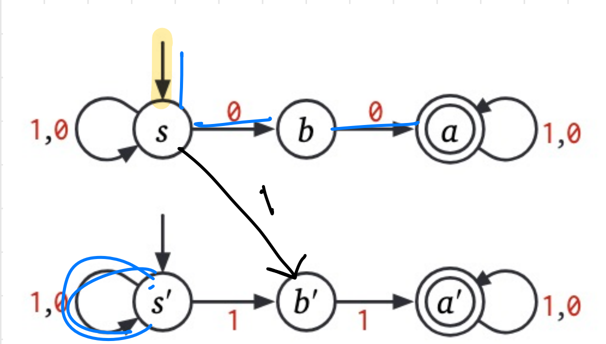
- what language does this NFA accept?

$(0+1)^* 1 (0+1)(0+1)$

- note that the smallest DFA for this language has 8 states. How would you prove this fact?

More NFA options:

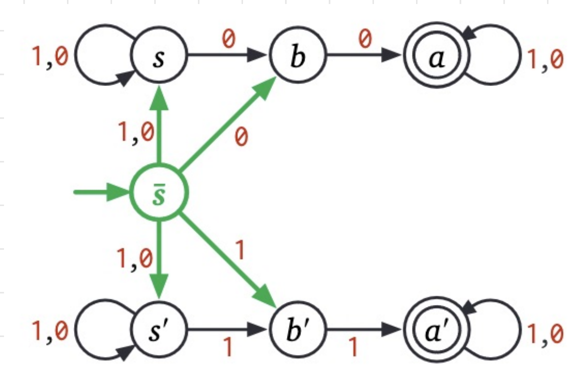
- multiple start states



$S = \{s, s'\}$

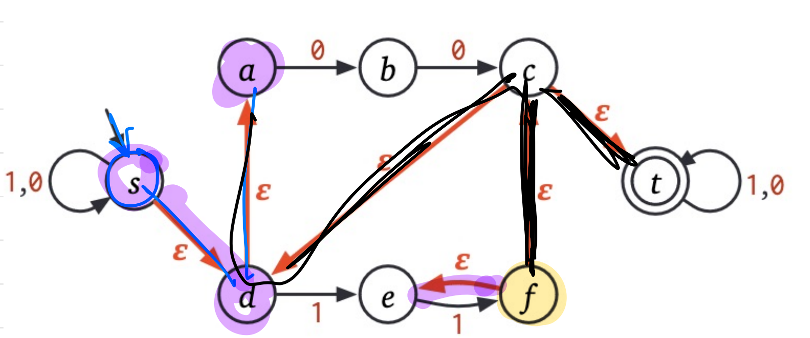
101
00

How can I transform NFA w/ multiple start states into an equivalent NFA w/ one start state?



ε-transitions

101

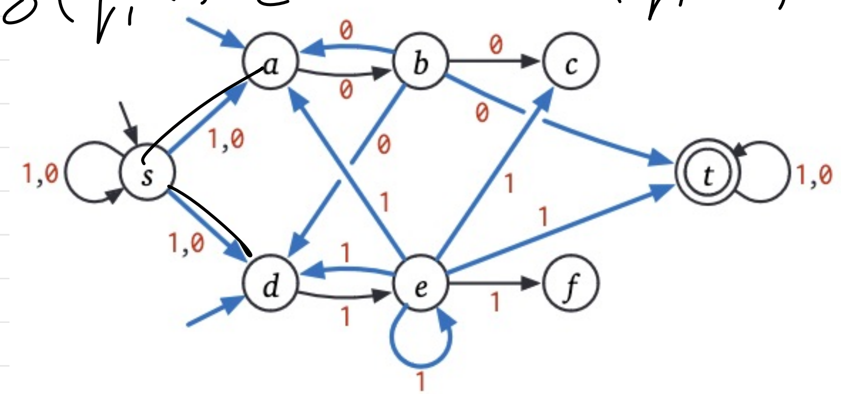


ε-reach(q): all states reachable from q by a sequence of ε-transitions

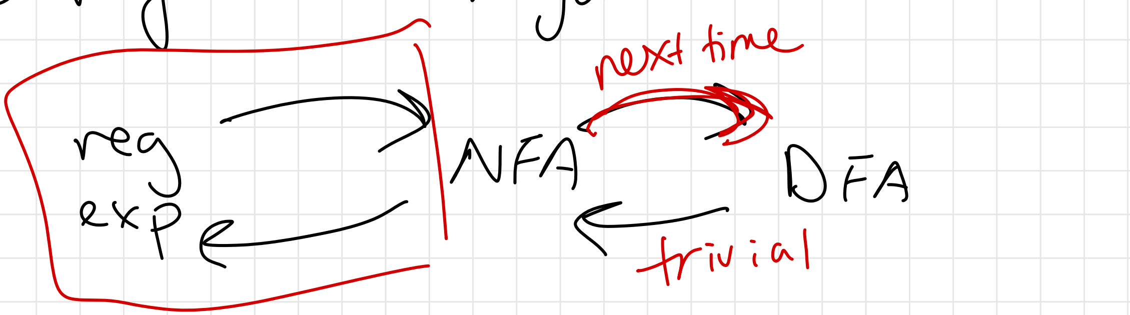
ε-reach$(f) = \{e, c, f, d, a\}$

To convert NFA M w/ ε-transitions to equivalent NFA M' without:

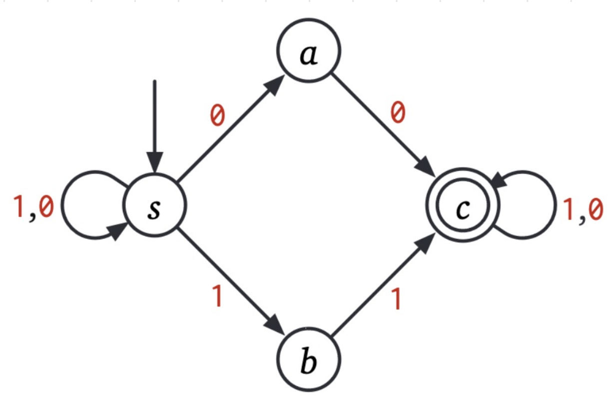
$Q' = Q$
$s' = \varepsilon\text{-reach}(s)$
$A' = A$
$\delta'(q,a) = \varepsilon\text{-reach}(\delta(q,a))$



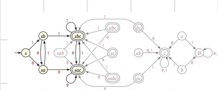
our goal: regular <=> automatic



NFA → DFA via subset construction



w = 01101001



Given NFA $M = (Q, s, A, \delta)$, define DFA M' as follows:

$Q' = 2^Q$

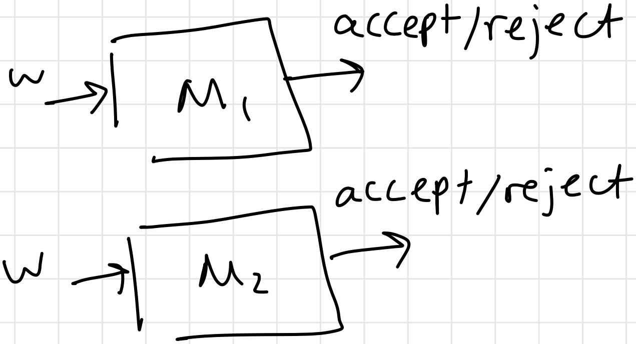
$s' = \{s\}$

$A' = \{q' \subseteq Q : q' \cap A \neq \emptyset\}$

$\delta'(q', a) = \bigcup_{q \in q'} \delta(q, a)$

Language Transformations

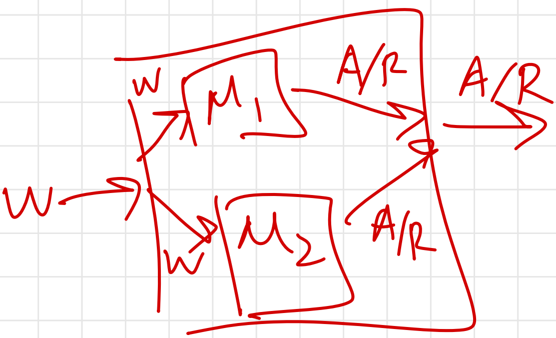
recall product construction:



$L_1 \cup L_2$

$L_1 \cap L_2$

$L_1 \oplus L_2$

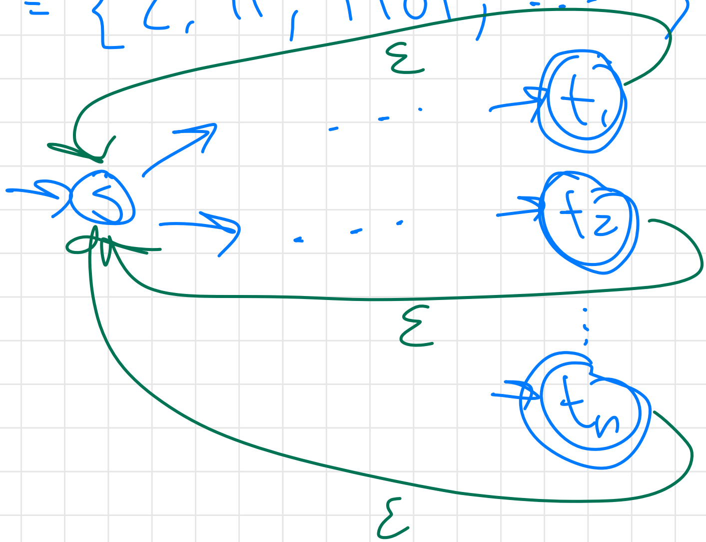


Suppose I have DFA M accepts L.

How do I build an NFA accepting L^*?

$L = \{\varepsilon, 1, 101\}$

$L^* = \{ \varepsilon, 11, 1101, \ldots \}$



DFA accepting L

let $flip(w) = \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ 1 \cdot flip(x) & \text{if } w = 0x \\ 0\, flip(x) & \text{if } w = 1x \end{cases}$