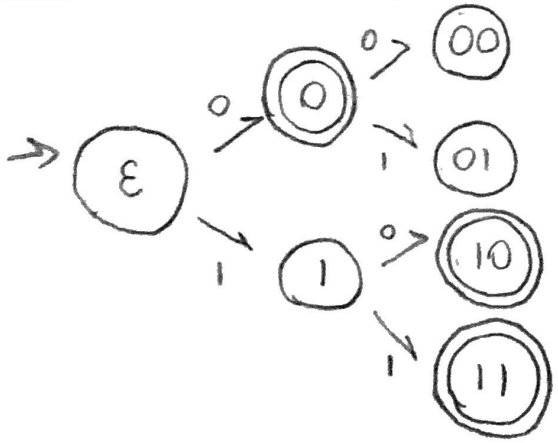
# CS 334: Problem Set 2

1. Every finite language is regular.

True, we can construct an fsa. s.t each possible combination of its alphabet is represented by a state, upto length n, which is the longest string in the language.

Ex: $A = \{0, 10, 11\}$ ← $n = 2$

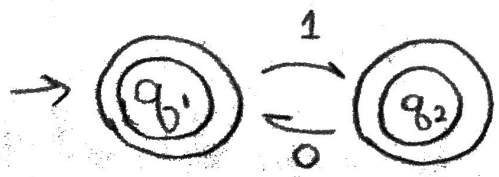



Harris Spahic
9/19/21
"I pledge my honor I have abided by the Steven's Honor System"

---

1b. If every state of an NFA is an accept state then its language is $\Sigma^*$.

False, Counter example.




If $a = 11$

$a \in \Sigma^*$

$a \notin F$

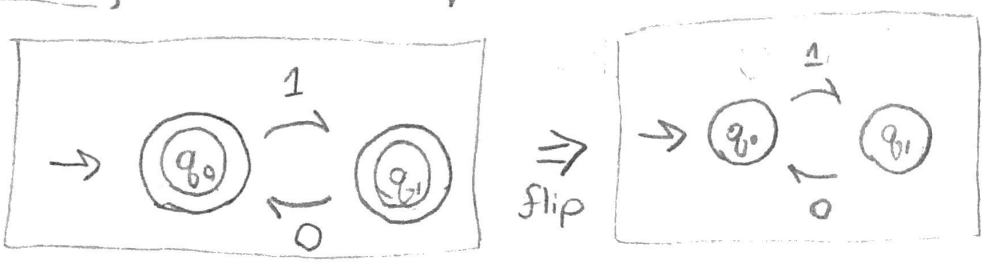
---

1c. If a k-state DFA accepts a string of length k, then its language is infinite.

True, since the initial state does not require a transition, there must exist a path containing the initial state, a closed cycle, & a final state. That is, k states at most k-1 transitions to unique nodes → closed cycle. As we saw from PS1, these conditions imply an infinite language.

1d. If every state of an NFA for language L is flipped then the new NFA accepts compliment of L.

False, counter example




$L = 1(01)^* \cup (01)^*$ $\quad L_{flip} = \emptyset \neq L^c$ (Example: $11 \notin L \rightarrow 11 \in L^c$)

1e. If every state of an NFA is flipped then the language accepted is not necessarily regular.

False, if we flip every state of an NFA we still have a valid NFA. Since NFA's only accept regular languages then the flipped NFA only accepts a regular language.

#2) Since we know DFA's only accept regular langnages it is sufficient to show we can convert any OFA into an equivalent DFA. In order to do this we convert the OFA into a DFA using the same method as an NFA (proof in book), however we redefine the final states of the resulting DFA to be:

(NFA) $F' = \{R \in Q' \mid R \text{ contains a accept state of } N\}$

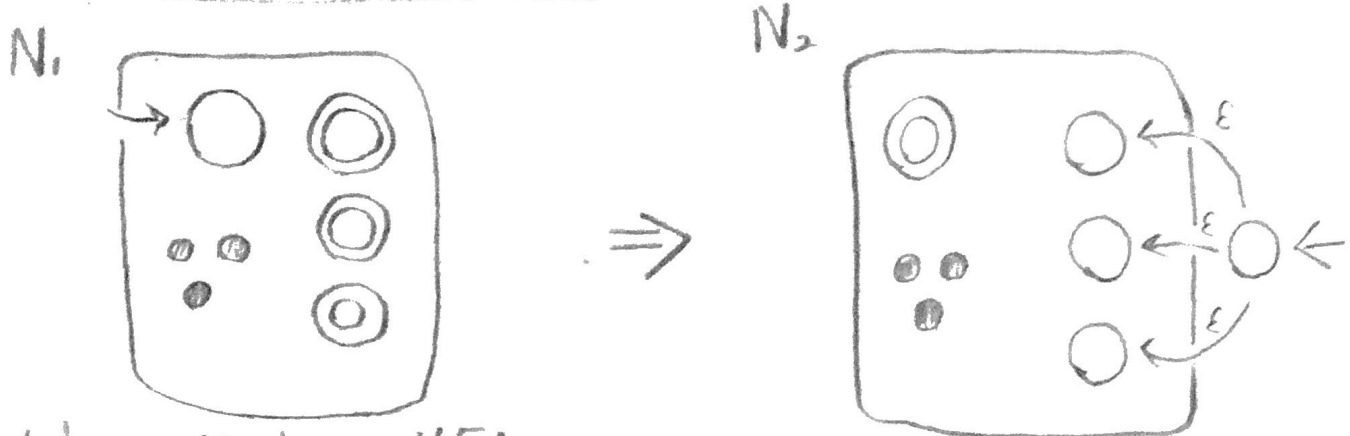
$\Downarrow\Downarrow$

(OFA) $F' = \{R \in Q' \mid R \text{ contains ALL accept states of } N\}$

→ Thus only the accept states of the resulting DFA change, proving we can construct an equivalent DFA & thus the OFA only accepts ∀ and is regular languages

#3) For any language A, let $A^R = \{w^R \mid w \in R\}$.
Show that if A is regular, so is $A^R$.

## Proof by construction




Where $N_1$ is an NFA with language A

Where $N_2$ is an NFA with language $A^R$

### Intuition

Basically start at all accept states & go reverse

### Proof

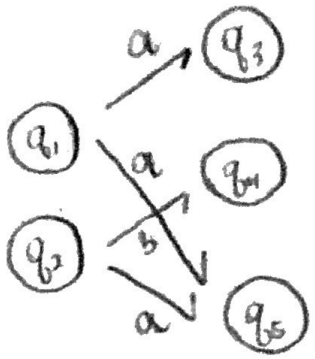
Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$. Construct $N_2 = (Q, \Sigma, \delta, q_0, F)$

1. $Q = \{q_0\} \cup Q_1$ → All states of $N_1$ + $q_0$
2. $q_0 = q_0$
3. $F = q_1$ → Accept state is start state of $N_1$
4. $\forall q \in Q$ & $\forall a \in \Sigma_\varepsilon$

Let $\delta^R(q,a) = \bigcup \{q_1 \text{ if } q_1 \in \delta(q,a)\} \cup \{q_2 \text{ if } q_2 \in \delta(q,a)\} \cup \cdots \cup \{q_n \text{ if } q_n \in \delta(q,a)\}$

$$\delta(q,a) = \begin{cases} \emptyset & q = q_0 \text{ \& } a \neq \varepsilon \\ F_1 & q = q_0 \text{ \& } a = \varepsilon \\ \delta_1^R(q,a) & q \in Q_1 \end{cases}$$

## Example of $\delta^R(q, a)$.




Then $\delta(q_1, a) = \{q_3, q_5\}$

$\delta(q_2, a) = \{q_5\}$

$\delta(q_2, b) = \{q_4\}$

$\rightarrow \delta^R(q_5, a) = (q_1 \text{ if } q_5 \in \delta(q_1, a))$

$\cup (q_2 \text{ if } q_5 \in \delta(q_2, a))$

$\vdots$

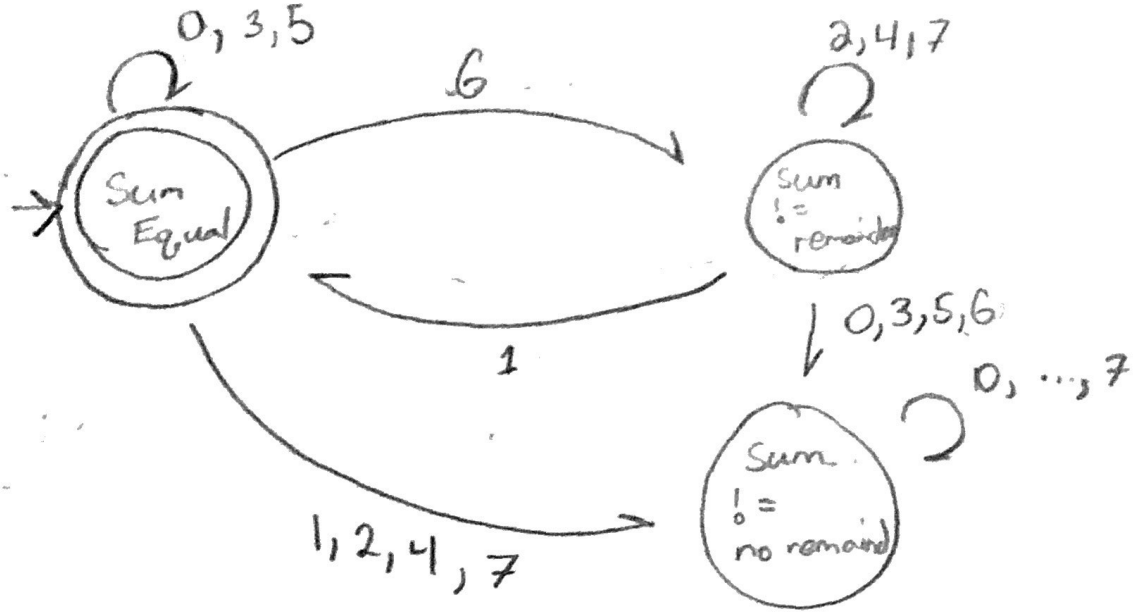
$= \{q_1, q_2\}$

$\delta^R(q, a)$ returns the set of nodes that map to $q$ following $a$.

---

In words we constructed a reverse NFA which automatically maps the start state to each accept state of the original. It then follows the input string in reverse from each state and accepts if this "backwards-walk" makes it to the start state of the original NFA. This will accept $A^R$. ▨

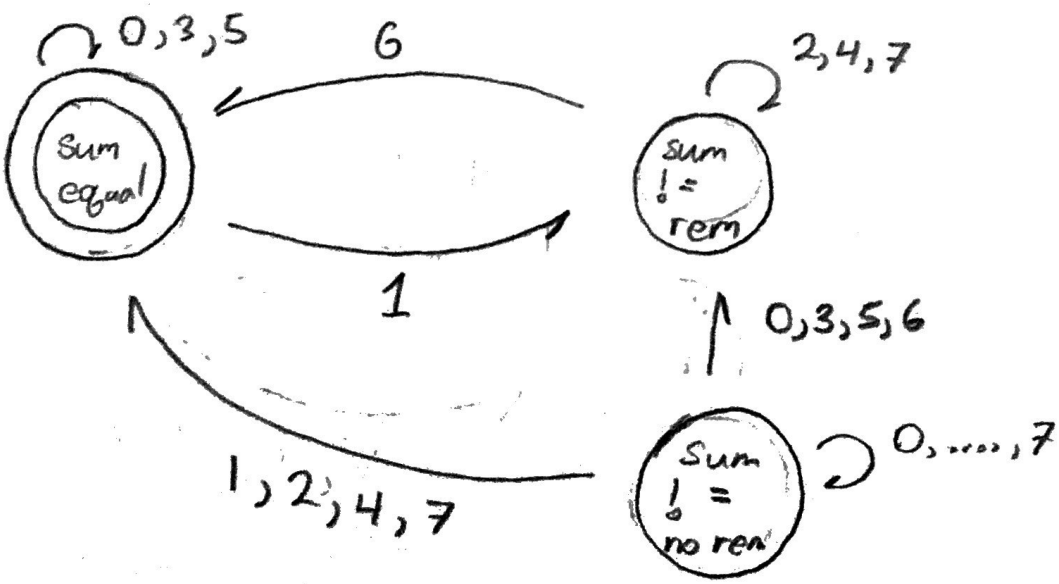
#4) Start with $B^R$ reverse.



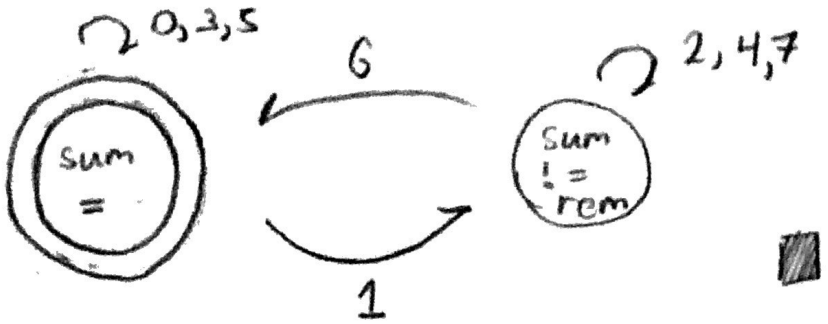

$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}$

$= \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix} \right\}$

Using the reversal process of #3 →




Which simplifies to




(This is beautiful. It's like doing all the work and always getting the most efficient result.)