CPSC 421/501 Sept 27, 2023

Feedback:

- Many thanks for the feedback!

- Terms: decidable, recognizable, etc. — there is a table in §2.4 of the handout

- Remarks on my enthusiasm and handwriting

- Next reason for enthusiasm: regular and non-regular languages over $\boxed{\Sigma = \{a\}}$.

New as of 2021, based on a question of Markus de Medeiros

Remark.

NON-PYTHON (in the handout)

equals

$\boxed{\text{NOT-PROG-PLUS-INPUT (in class)}}$,

equals

$\{$ strings not of the form $p \, \sigma_0 \, i$ where p is a valid Python program $\}$

is decidable (in polynomial time)

Also

GROUCHO-MARX-SELF equals

NON-SELF-ACCEPTANCE equals

$$\{ \, p \mid p \notin \text{LanguageRecBy}(p) \, \}$$

Today

- Review the formal definition of a DFA as a tuple $(Q, \Sigma, \delta, q_0, F)$

- Possible examples:

$$L = \left\{ w \in \{a,b\}^* \;\middle|\; \begin{array}{l} \text{first letter of } w \\ = \text{last} \quad " \quad " \quad " \end{array} \right\}$$

$$L = \left\{ w \in \{0,1\}^* \;\middle|\; \begin{array}{l} w \text{, in binary, represents} \\ \text{an integer divisible} \\ \text{by } 3 \end{array} \right\}$$

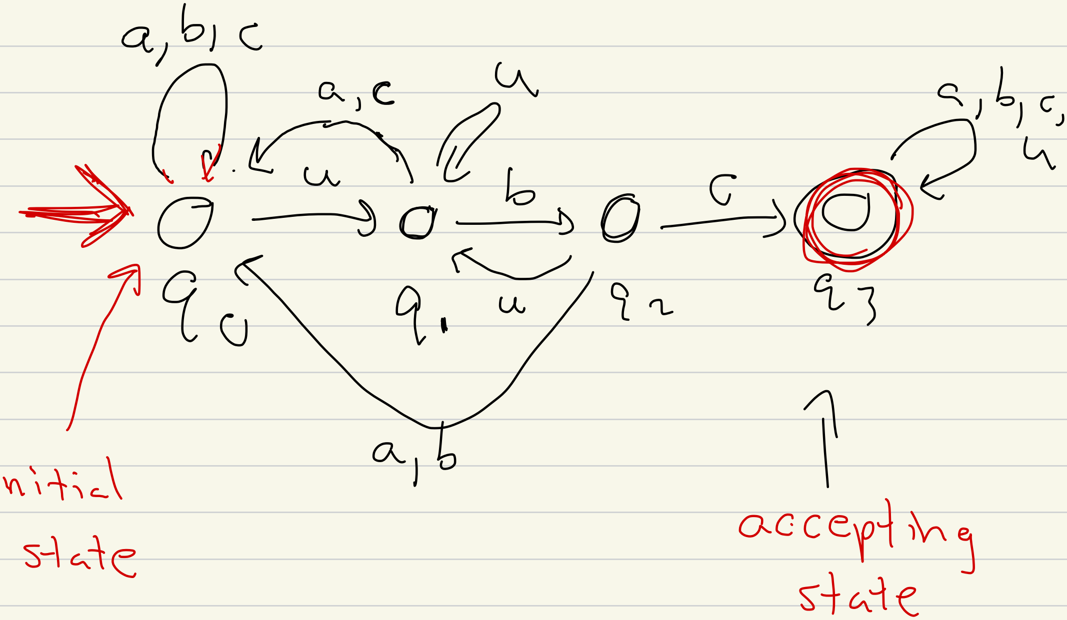
and/or similarly, with $w \in \{0,1,\ldots,9\}^*$, etc.

- Regular and non-regular languages when $\Sigma = \{a\}$

Last class: begin with example

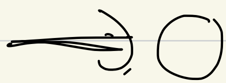
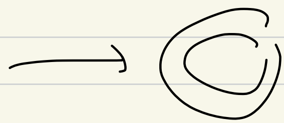
$\Sigma = \{a, b, c, u\}$

$L = \{ s_1 \underline{ubc} s_2 \mid s_1, s_2 \in \Sigma^* \}$

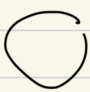


Q = set of states = $\{q_0, q_1, q_2, q_3\}$

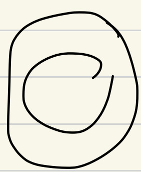
In the picture of what a DFA is

 or 

<u>initial state</u>



non-accepting state



accepting state

input $\sigma_1 \sigma_2 \sigma_3 \cdots \sigma_n$

Formally, a DFA (deterministic finite automaton) is a tuple

$$(Q, \Sigma, \delta, q_0, F)$$

Q = states of DFA

Σ = alphabet

$$\delta : Q \times \Sigma \longrightarrow Q$$

meaning

$\delta(q, \sigma)$ = the next state that you move to when in state q, next input symbol is σ

q_0 = initial state

F = set of accepting states

= " " final "

We say a DFA "$(Q, \Sigma, \delta, q_0, F)$ accepts $\bar{i}$" if

$\bar{i} \in \Sigma^*$, and when we run $\bar{i}$ on $M = (Q, \Sigma, \delta, q_0, F)$

we end in a state $\in F$.

<u>Language</u> <u>recognized</u> <u>by</u> M

is

$$\{ i \in \Sigma^* \mid M \text{ accepts } i \}$$

A language $L \subset \Sigma^*$ is **regular** if L is recognized by some DFA, $M = (Q, \Sigma, \delta, q_0, F)$

(otherwise we say L is non-regular)

make the formalities as simple as possible

Remark

$\bigcirc \longleftrightarrow$ "no"

$\circledcirc \longleftrightarrow$ "yes"

all DFA's {can be viewed as} very simple Python programs

Q in a DFA roughly $\longleftrightarrow$ program line or set of lines

"Complexity" of a regular might be the minimum number of states

Another example:

$$\left\{ s \in \{0,1,\ldots,9\}^* \;\middle|\; \begin{array}{l} s \text{ represents a} \\ \text{string in decimal} \\ \text{divisible by } 3 \end{array} \right\}$$

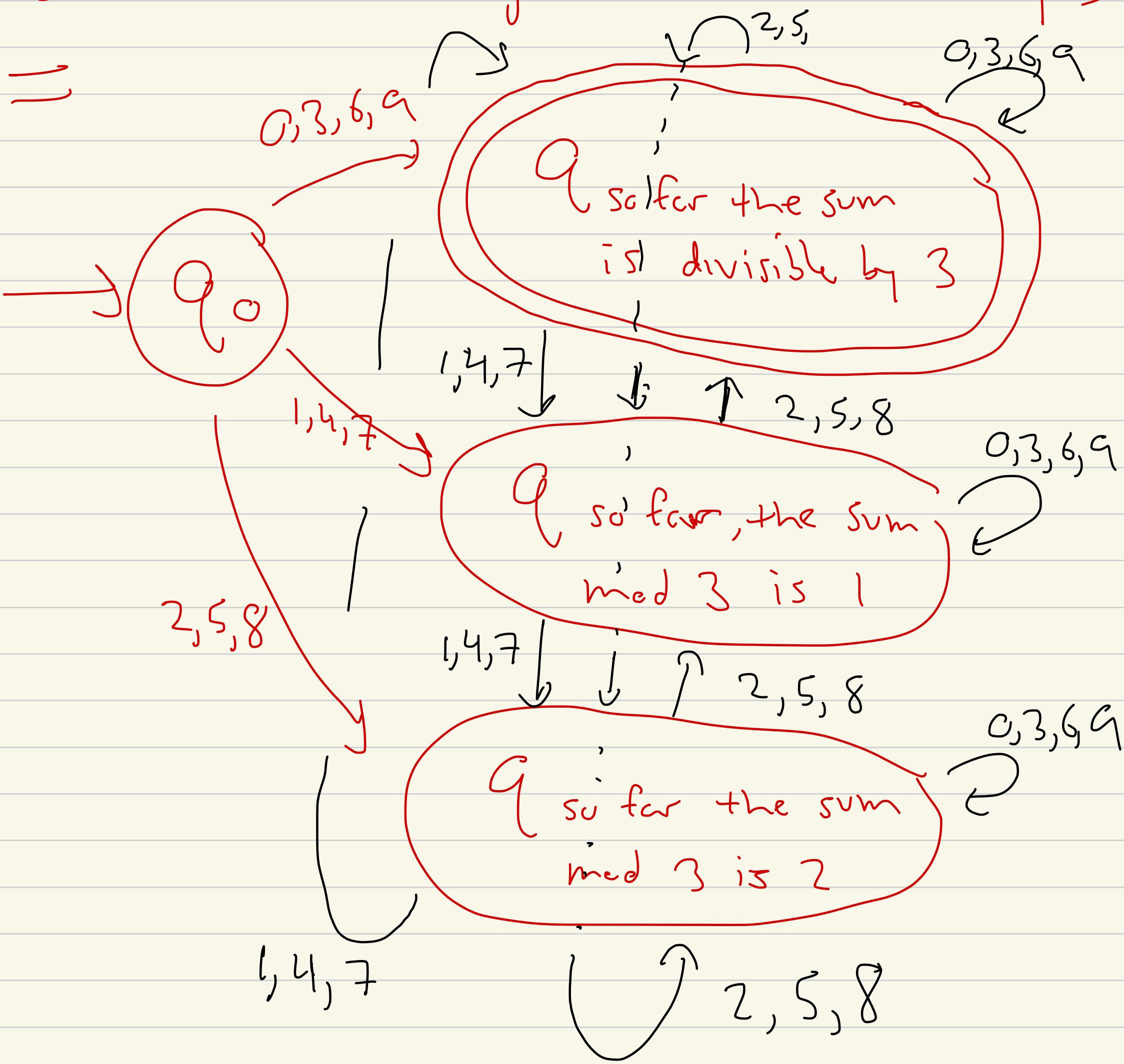
- we don't allow ε
- leading 0's OK

$$= \{ 0, \overset{3,6,9}{0}0, 03, 06, 09, 12, 15, \ldots$$
$$96, 99, 000, 003, \ldots \}$$

DIV-BY-3-IN-DECIMAL-LEADING-0's-OK

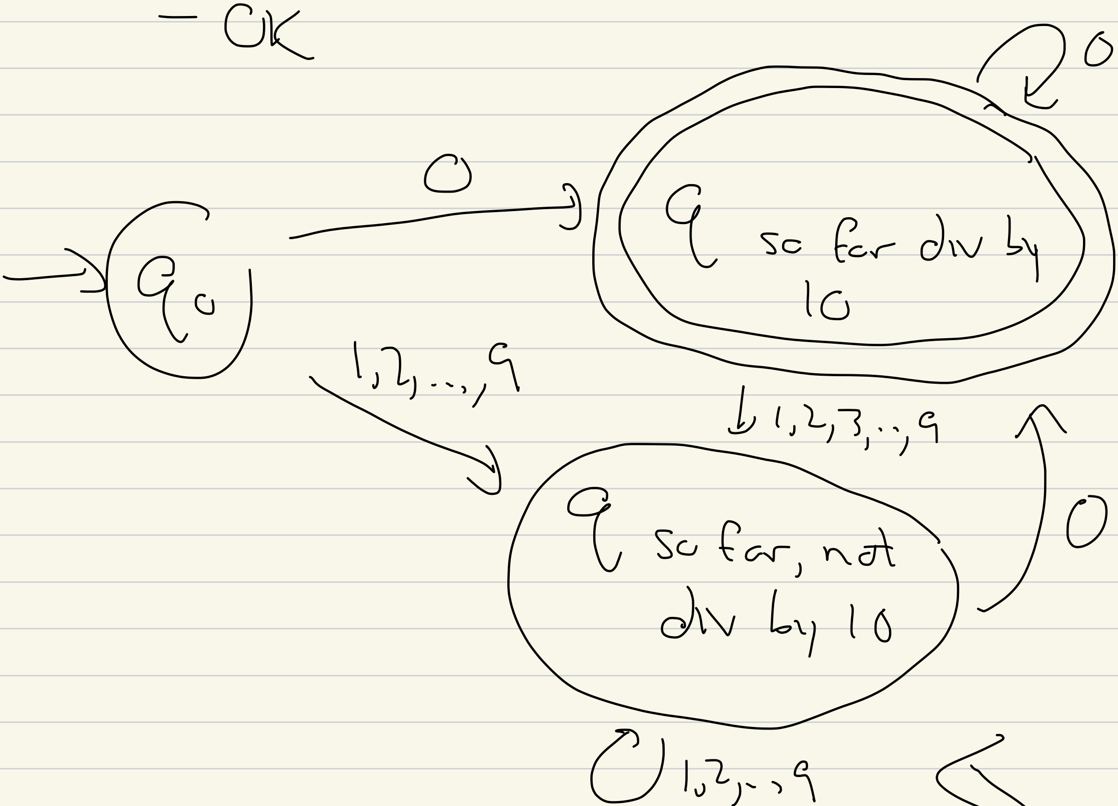
Algorithm: an integer in decimal is divisible by 3 iff the sum of its digits is divisible by 3

$\Longrightarrow$

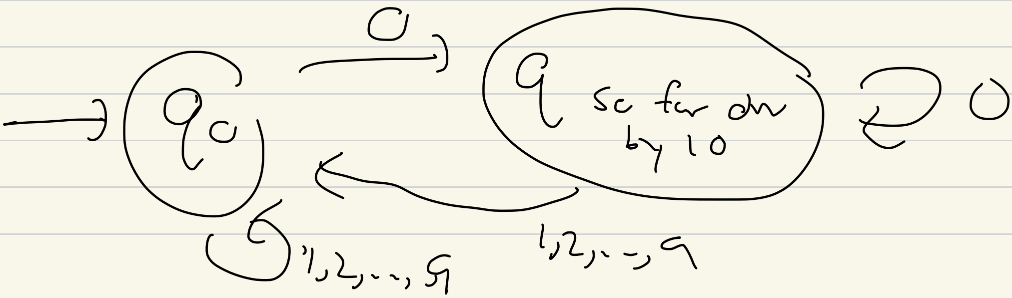


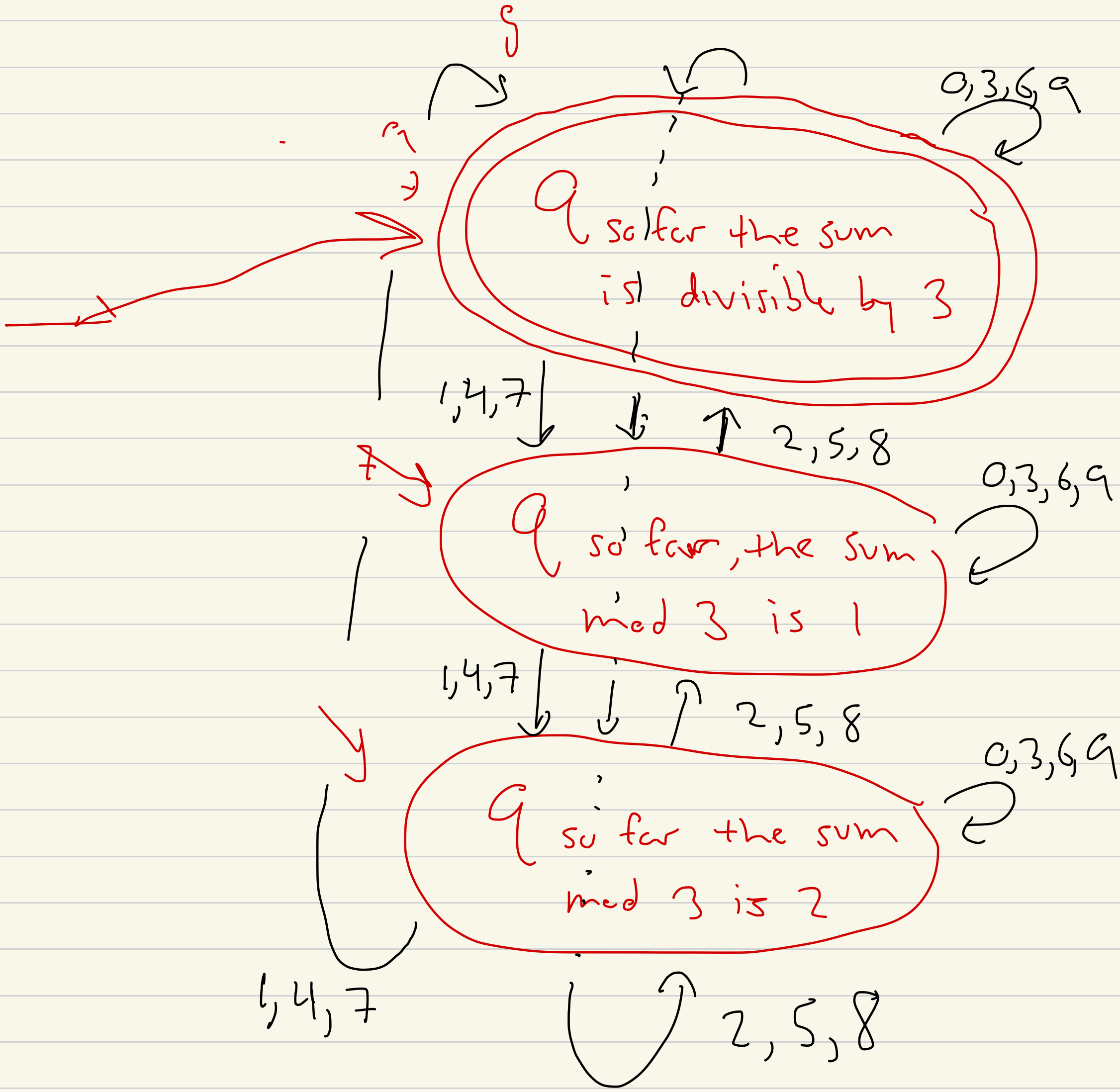
DIV-BY-10-IN-DECIMAL-LEADING-0's

- OK



We can merge q_0 and




q so far the sum is divisible by 3
0,3,6,9
1,4,7
2,5,8
q so far, the sum mod 3 is 1
0,3,6,9
1,4,7
2,5,8
q so far the sum mod 3 is 2
0,3,6,9
1,4,7
2,5,8