Chapter 1: Regular Languages

→ Section 1.1 Finite automata (DFA's)
Section 1.2 Non-deterministic finite automata (NFA's)
Section 1.3 Regular Expressions ⟵ languages
Section 1.4 Proving certain languages are not regular

Very very simple algorithms

- Real story 1: DFA's & NFA's are very simple algorithms

- Real story 2: grep, egrep (extended grep), looking in many long text files, searching for the occurrence of certain substrings, Unix/Linux builds NFA or DFA

Finite automata:

Consider some simple languages, e.g.

$$\text{DIV\_BY\_2} = \{ s \in \{0,1,\ldots,9\}^* \mid s \text{ represents an integer divisible by } 2 \}$$

leading 0's not OK

empty string, ε, not OK

$0, 2, 4, 117396, 12 \in \text{DIV\_BY\_2}$

$13, 121, 012, \varepsilon \notin \text{DIV\_BY\_2}$

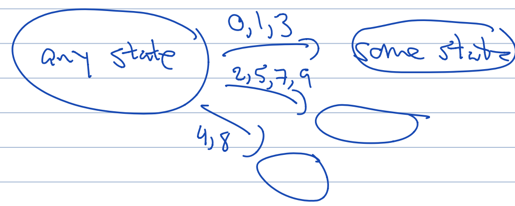
Algorithm: Is 1234567809312 in DIV_BY_2 ?

Intuitively: build a simple, left-to-right scan of

1 2 3 4 5 6 7 8 0 9 3 1 2

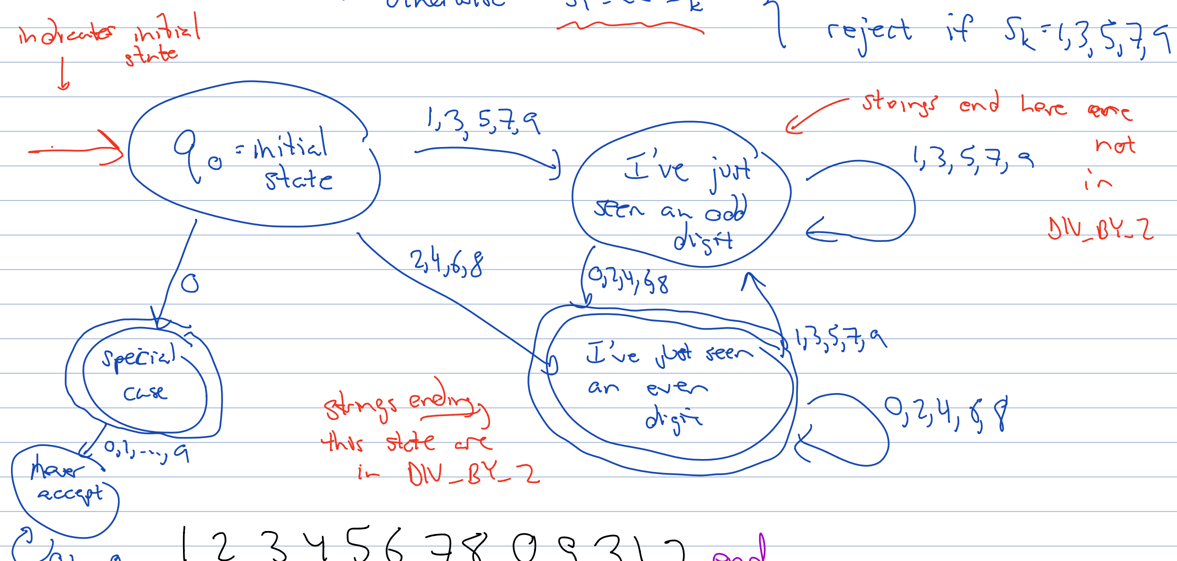
start

Finite automaton has: finite set of states, Q,
involves finite alphabet, Σ,



Example:

DIV_BY_2 :
- reject ε
- reject 0 other stuff
- accept 0
- otherwise $s_1 \dots s_k$ $\begin{cases} \text{accept if } s_k = 0,2,4,6,8 \\ \text{reject if } s_k = 1,3,5,7,9 \end{cases}$

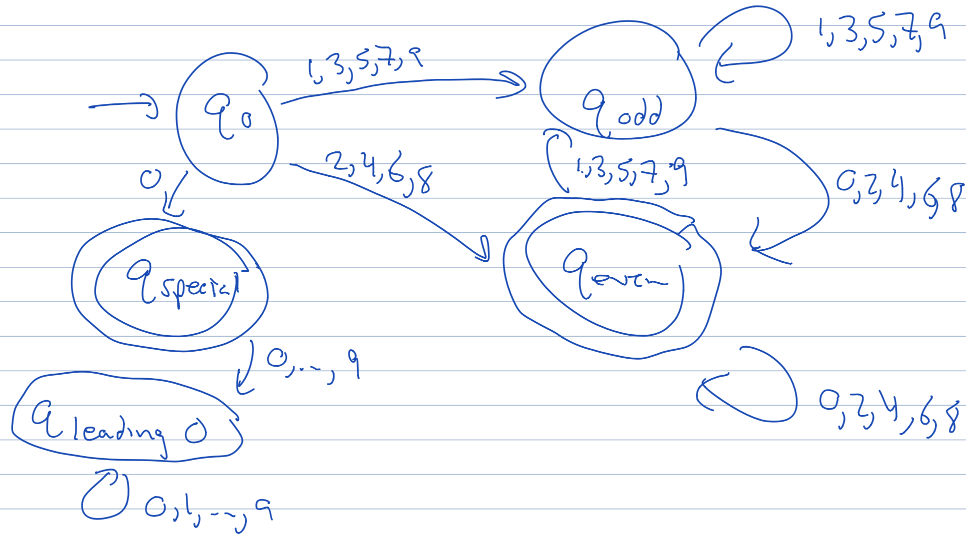


1 2 3 4 5 6 7 8 0 9 3 1 2 end

start

Special cases: $\varepsilon \leftrightarrow$ reject $\quad$ $0 \leftrightarrow$ accept

$0\, s_2 \dots s_k \leftrightarrow$ reject



Formally: states Q, Σ = alphabet, $\delta : Q \times \Sigma \to Q$

q_0 = initial state

$\delta(q, \sigma)$ = the state that you move to from q seeing the symbol σ

"accepting state" "final states" F

Formally: a deterministic finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ s.t. ~~~~~ [legal document]

DIV_BY_3 = $\{ s \in \{0,1,\dots,9\}^* \mid s$ represents an integer divisible by 3 $\}$

PRIMES = $\{$ " $\mid s$ represents a prime number $\}$

We say that a language is <u>regular</u> if there is a DFA recognizes the languages

E.g. DIV_BY_2, DIV_BY_3, are regular } To show regular, need to build a DFA that recognizes it

SQUARES, PRIMES, is not regular } More difficult to show

DIV_BY_3: DFA: idea: see if what we have so far ~~is~~ (1) is divisible by 3

(2) mod 3 is 1

(3) mod 3 is 2