Finite Automata Overview

DFA Deterministic Finite Automata

recognizes (computes) regular languages (RL's)

NFA Nondeterministic Finite Automata

Arise from proving closure properties of RL's

Similar to DFA's, but allowed to make "educated guesses"

Regular Expressions Define languages via closure properties

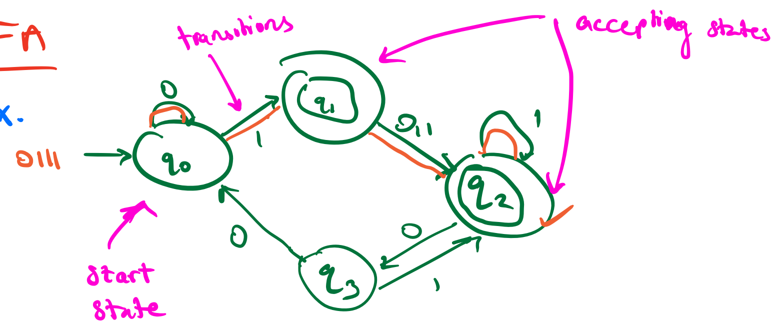
RE's recognize RL's, so do DFA + NFA

Later topics of focus

- Pumping Lemma
- Myhill-Nerode Theorem
- Streaming algorithms
- Communication complexity
- Randomness

DFA

ex.



Automaton accepts string if end on accepting state.
Otherwise, it rejects.

Why study DFA's?

→ Constant-size memory

→ Read-once input

→ Relevant to streaming algos which can only read once

vocabulary

alphabet Σ: finite set (i.e., $\Sigma = \{0, 1\}$)

string over Σ: finite length sequence of elements of Σ

Σ^*: set of all strings over Σ

For string x: $|x|$ length of x

ε: empty string

Language over Σ: set of strings over Σ (subset of Σ^*)

OR boolean function over strings

Languages L correspond to functions

$f: \Sigma^* \to \{0, 1\}$

where $f(x) = 1$ if $x \in L$ else 0

Formalization

DFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

where

Q: set of states (finite)

Σ: alphabet (finite)

$\delta: Q \times \Sigma \to Q$: transition fn

$q_0 \in Q$: start state

$F \subseteq Q$: set of accepting states

M accepts string $w = w_1 \dots w_n \in \Sigma^*$

if $\exists\, r_0 \dots r_n \in Q$ where

$r_0 = q_0$

$\delta(r_i, w_{i+1}) = r_{i+1} \quad i = 0 \dots n-1$

$r_n \in F$

$L(M)$: set of strings accepted/recognized by M

Regular Languages

Language L' regular if

L' recognized by a DFA

$(\exists$ DFA M where $L' = L(M))$

Closure properties of RL's

Union: $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$

Intersection: $A \cap B = \{w \mid w \in A \text{ or } w \in B\}$

Complement: $\neg A = \{w \in \Sigma^* \mid w \notin A\}$

Reverse: $A^R = \{w_1 \dots w_k \mid w_k \dots w_1 \in A, w_i \in \Sigma\}$

Concatenation: $A \cdot B = \{vw \mid v \in A, w \in B\}$

Star: $A^* = \{s_1 \dots s_k \mid k \geq 0, s_i \in A\}$

If A, B regular, so are

$A \cup B$

$A \cap B$

$\neg A$

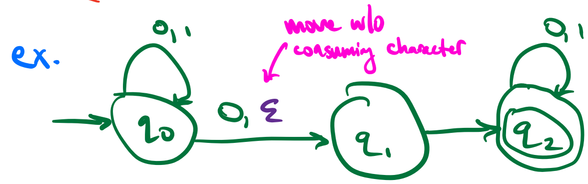
A^R $\Leftarrow$ this proof leads to define NFA's

$A \cdot B$

A^*

NFA's

ex.



accepts strings that contain 0

Allows multiple start states

(can use ε transitions to reduce to single start state)

formalization

NFA is a 5-tuple $N = (Q, \Sigma, \delta, Q_0, F)$

where

Q : set of states

Σ : alphabet

$\delta : Q \times \Sigma_\varepsilon \to 2^Q$ transition function

augment ε to Σ

set of all subsets of Q

$Q_0 \subseteq Q$: set of start states

$F \subseteq Q$: set of accepting states

N accepts $w = w_1 \cdots w_n \in \Sigma^* \cup \{\varepsilon\}$

if $\exists\, r_0 \cdots r_n \in Q$

where

$r_0 \in Q_0$

$r_{i+1} \in \delta(r_i, w_{i+1})$ $\quad i = 0 \ldots n$

$r_n \in F$

differences w/DFA

→ generally NFA much simpler

→ DFA accepts or rejects; NFA only needs one path to accept in order to accept

NFA - DFA equivalence

NFA's recognize only regular languages too!

Converting NFA to DFA

Input: NFA $N = (Q, \Sigma, \delta, Q_0, F)$

Output: DFA $D = (Q', \Sigma, \delta', q_0', F')$

To learn if NFA accepts, do computation in parallel - maintain set of all possible states that could have been reached

let $Q' = 2^Q$

let $\delta' : Q' \times \varepsilon \to Q'$

reading ε's   reading chars

where $\delta'(R, \sigma) = \bigcup_{r \in R} \varepsilon(\delta(r, \sigma))$

let $q_0' = \varepsilon(Q_0)$

let $F' = \{R \in Q' \mid f \in R \text{ for some } f \in F\}$

⇒ done!

Regular Expressions

Defines computation as a simple, logical description of RL's

Inductive definition

Let Σ alphabet

For all $\sigma \in \Sigma$, σ is a regexp (representing $\{\sigma\}$)

ε is a regexp (representing $\{\varepsilon\}$)

$\emptyset$ is a regexp (representing $\emptyset$)

If R_1, R_2 are regexps for L_1, L_2:

$(R_1 R_2)$ representing $L_1 \cdot L_2$

$(R_1 + R_2)$ representing $L_1 \cup L_2$

$(R_1)^*$ representing L_1^*

are regexps

Precedence: $*$, then $\cdot$, then $+$

Example

Let $\Sigma = \{0, 1\}$

Let $L = \{w \mid w \text{ has exactly one } 1\}$

Then $R = 0^* 1 0^*$

the single 1

any number of 0's

DFAs, NFAs, and regexp's are equivalent

Proof: induction.