0. Introduction

0.1. Automata, Complexity and Computability

Complexity theory - What makes problems computationally hard?
Computability theory - What problems are computable?
Automata theory - definitions and properties of mathematical models of computation

0.2. Strings and Languages

alphabet - nonempty finite set of symbols (Σ, Γ)
string over alphabet - finite sequence of symbols from alphabet
$w \Rightarrow |w|$ (lenght) - number of symbols
ε - empty string $\Rightarrow |\varepsilon| = 0$
w^R (reverse) - reverse order of symbols of w
z is substring of w - z appears consecutively in w
$|x| = n$, $|y| = m \Rightarrow xy$ - concatenation, $|xy| = n+m$
Lexicographic order - alphabet/dictionary order
shortlex/string order - alphabet order but mainly length order
x - prefix of y if $\exists z$ s.t. $xz = y$, x - proper prefix, $x \neq y$
language - set of strings
prefix-free language - no member is proper prefix ~~of~~ another

1. Deterministic Finite Automaton (DFA)

finite automaton - simplest model of computation (limited memory)
States = Nodes, Starting/Initial State ($\rightarrow$O), Accepting/Final state (◎)
Transitions = Edges, Alphabet of Symbols (Σ)
FSM $\rightarrow$ generate string = traverse graph
↳ accept/recognize string = does graph traversal by string end up in accepting node?

<u>Formal Definition</u> of DFA, $M = (Q, \Sigma, \delta, q_0, F) \rightarrow$ 5-tuple

- $\rightarrow Q$ - Set of states } finite
- $\rightarrow \Sigma$ - Alphabet, set of Symbols }
- $\rightarrow \delta$ - transition function; $\delta: Q \times \Sigma \rightarrow Q$
- $\rightarrow q_0$ - starting/initial state; $q_0 \in Q$
- $\rightarrow F$ - Set of accepting/final states; $F \subseteq Q$

2. Nondeterministic Finite Automaton (NFA)

Given the current state, there may be multiple next states.

If there is any way to run the machine that ends with ACCEPT, then NFA accepts

Formal Definition of NFA, $M=(Q,\Sigma,\delta,q_0,F)$, $\Sigma_\varepsilon=\Sigma\cup\{\varepsilon\}$, $\delta: Q\times\Sigma_\varepsilon\to P(Q)$

Powerset - set of all subsets (incl. $\varnothing$ and full set) $\Rightarrow |\Sigma|=n, |P(\Sigma)|=2^n$

Th For every NFA there is equivalent DFA and vice versa.

Proof by construction

Let $M=(Q,\Sigma,\delta,q_0,F)$, NFA.

Construct $M'=(Q',\Sigma,\delta',q_0',F')$, DFA s.t. $Q'=P(Q)$; $F'=\{R\in Q' \mid R$ contains an accept state from NFA$\}$

$\delta'(R,a)=\{q\in Q \mid q\in E(\delta(r,a))\}$, $q_0'=E(q_0)$

$E(R)=\{q\in Q \mid q$ can be reached from state in R following ε-edges$\}$

3. Operations on Regular Languages

Def. Regular language - language recognized by some finite automaton

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

→Concatenation: $A\circ B=\{xy \mid x\in A \text{ and } y\in B\}$

→Star: $A^*=\{x_1x_2...x_k \mid k\geq 0 \text{ and each } x_i\in A\}$

Th Class of Regular Languages is closed under union.

L_1, L_2 - regular languages $\Rightarrow L_1\cup L_2$ is regular language.

Proof by construction (DFA)

Assume $L_1=L(M_1)$, $L_2=L(M_2)$, $M_1=(Q_1,\Sigma_1,\delta_1,q_1,F_1)$, $M_2=(Q_2,\Sigma_2,\delta_2,q_2,F_2)$

Build M to recognize $L_1\cup L_2$.

Simulate M_1 and M_2 simultaneously $\Rightarrow$ state in M corresponds to 2 states

$M=(Q,\Sigma,\delta,q_0,F)$ s.t. $Q=Q_1\times Q_2=\{(r_1,r_2) \mid r_1\in Q_1 \text{ and } r_2\in Q_2\}$; $\Sigma=\Sigma_1\cup\Sigma_2$;

$\delta((r_1,r_2),a)=\delta_1(r_1,a),\delta_2(r_2,a)$; $q_0=(q_1,q_2)$; $F=\{(r_1,r_2) \mid r_1\in F_1 \text{ or } r_2\in F_2\}$

Proof by construction (NFA)

Assume $L_1=L(M_1)$, $L_2=L(M_2)$, $M_1=(Q_1,\Sigma_1,\delta_1,q_1,F_1)$, $M_2=(Q_2,\Sigma_2,\delta_2,q_2,F_2)$

Construct $M=(Q,\Sigma,\delta,q_0,F)$ s.t. $Q=Q_1\cup Q_2\cup\{q_0\}$; $\Sigma=\Sigma_1\cup\Sigma_2$; $F=F_1\cup F_2$;

$$\delta(q,a)=\begin{cases}\delta_1(q,a) & \text{if } q\in Q_1\\ \delta_2(q,a) & \text{if } q\in Q_2\\ \{q_1,q_2\} & \text{if } q=q_0 \text{ and } a=\varepsilon\\ \{\} & \text{if } q=q_0 \text{ and } a\neq\varepsilon\end{cases}$$

Th Class of Regular Languages is closed under concatenation
L_1, L_2 - regular languages $\Rightarrow L_1 \circ L_2$ is regular language.
Proof by construction (NFA)
Assume $L_1 = L(M_1)$, $L_2 = L(M_2)$, $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$, $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$
Construct $M = (Q, \Sigma, \delta, q_0, F)$ s.t. $Q = Q_1 \cup Q_2$; $\Sigma = \Sigma_1 \cup \Sigma_2$; $q_0 = q_1$; $F = F_2$;

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 \text{ and } q \notin F_1 \\ \delta_2(q,a) & \text{if } q \in Q_2 \\ \delta_1(q,a) \cup \{q_2\} & \text{if } q \in F_1 \text{ and } a = \varepsilon \\ \delta_1(q,a) & \text{if } q \in F_1 \text{ and } a \neq \varepsilon \end{cases}$$

Th Class of Regular Languages is closed under star.

4. Regular Expressions

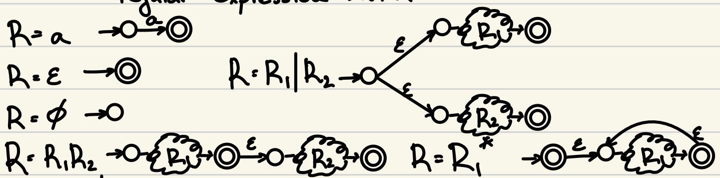
Def. Regular Expression — $a \in \Sigma$, $R_1 \cup R_2$ $(R_1 | R_2)$, $R_1 \circ R_2$ $(R_1 R_2)$, R_1^*, ε, $\emptyset$, (R_1)
where R_1, R_2 are regular expressions as well

Star Binds Tightest: $ab^* = a(b^*)$
Concatenation Binds Tighter Than Union: $ab \cup c = (ab) \cup c$, $ab|c = (ab)|c$
$a^* = \{a\}^* = \{a\}^*$; $a^+ = aa^* = \{a\}^+$; $[a] = a|\varepsilon = (a \cup \varepsilon) = a?$
$L(a) = \{a\}$, $L(R_1|R_2) = L(R_1) \cup L(R_2)$; $L(R_1 \circ R_2) = L(R_1) \circ L(R_2)$; $L(R_1^*) = L(R_1)^*$;
$L(\varepsilon) = \{\varepsilon\}$, $L(\emptyset) = \{\}$, $L((R_1)) = L(R_1)$; concatenate $\emptyset = \{\}$; $\emptyset^* = \{\varepsilon\}$

Th A language is regular iff some regular expression describes it.

Lemma Language described by regular expression $\rightarrow$ regular

Proof $\rightarrow$ closure of $\cup$, $*$ and $\circ$
$\rightarrow$ regular expression $\rightarrow$ NFA



Lemma Regular Language $\rightarrow$ can be described by regular expression

Proof

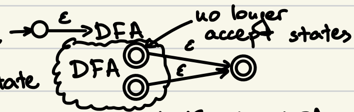
Build DFA that recognizes language.
Build GNFA (Generalized NFA) and reduce it.

Def. GNFA is NFA except:

→ edges labeled with regular expressions
→ only 1 accept state
→ exactly one edge between any states
→ no edges going to start state
→ no edges going out of accepting state

<u>DFA → GNFA</u>

add start state
add new accept state



eliminate multiple edges with UNION   DFA: a, b, c   GNFA: a|b|c
add missing edges with $\emptyset$

Reduced GNFA: $\to \circ \xrightarrow{\text{<regular expression>}} \circledcirc$

$\varepsilon R = R\varepsilon = R = \emptyset R = \{\}\cup R, \quad \emptyset R = R\emptyset = \emptyset$

5. Pumping Lemma

A - regular language → $\exists p$ s.t. any string s ($|s| \geq p$) can be divided into $s = xyz$ s.t.
1) $xy^iz \in A, \forall i \geq 0$ 2) $|y| > 0$ 3) $|xy| \leq p$

$B = \{0^n 1^n \mid n \geq 0\}$ → non-regular language

<u>Proof by contradiction</u>

Assume B is regular. Let p be the pumping length of B. Let $s = 0^p 1^p$.
Divide s into xyz:
case 1: y in zeros ⇒ xy^iz in B, but $\#0 \neq \#1$ (infinite 0's)
case 2: y in ones ⇒ xy^iz in B, but $\#0 \neq \#1$ (infinite 1's)
also violate 3) $|xy| \leq p$
case 3: y has zeros and ones ⇒ xy^iz in B, but there are 0's after 1's