1 Finite memory

1.1 DFA

Deterministic Finite Automaton, $M = (Q, \Sigma, \delta, q_0, F)$

Σ : alphabet, Q : finite set of States (independent form input length), $\delta : Q \times \Sigma \to Q$: transition function (arrows), $q_0 \in Q$ start state (entering arrow), $F \subset Q$ set of accepting state(s) (double circled)

Language of machine M : set A of all accepted strings, $L(M) = A$

Accepting state : validate input if reached

Regular language : Σ^* set of all strings composed by Σ (including ε), $L(M) \subseteq \Sigma^*$, $\exists$ DFA s.t. $L = L(M)$

Base case : prove for $w = \varepsilon$

Inductive case : assume $\delta(q_0, x) = q_i$ if $x \in T_i\ \forall i$, to prove : for each $\sigma \in \Sigma\ \delta(q_0, x.\sigma) = q_i$ if $x.\sigma \in T_i\ \forall i$, proof by case on what σ and $\delta(q_0, x)$ are

Complement : $\bar{L} = \{w \in \Sigma^* : w \notin L\}$

L regular $\implies \bar{L} = L(M')$ regular $(M' = (Q, \Sigma, \delta, q_0, \bar{F} = Q \setminus F))$

Union : $L_1 \cup L_2 = \{w \in \Sigma^* : w \in L_1$ or $w \in L_2\}$, $M = (Q = Q_1 \times Q_2, \Sigma = \Sigma_1 \cup \Sigma_2, \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta(q_2, a)), q_0 = (q_{1_0}, q_{2_0}), F = (q_1, q_2) : q_1 \in F_1$ or $q_2 \in F_2)$, (accept if one accept)

Interesection : $L_1 \cap L_2 = \{w \in \Sigma^* : w \in L_1$ and $w \in L_2\}M = (Q = Q_1 \times Q_2, \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)), q_0 = (q_1, q_2), F = \{(q_1, q_2) : q_1 \in F_1$ and $q_2 \in F_2\})$, (accept if both accept)

Concatenation : $L_1 \circ L_2 = \{w \in \Sigma^* : w = w_1.w_2, w_1 \in L_1$ and $w_2 \in L_2\}$

1.2 NFA

Parallel computer, transition to $\geq$ 1 state on symbol, state may have 0 transition on symbol, take step without reading symbol ε-transitions, $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$

Accepts : $\exists$ choices that accepts (read full sequence)

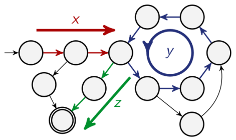
$\forall$ NFA $\exists DFA \equiv NFA$, language regular $\iff \exists$ NFA recognize it

Delta : $\Delta : Q \times \Sigma^* \to Q$: apply FA to seq.; $Q_M = 2^{Q_N}, \forall A \subseteq Q_N, x \in \Sigma^*, \Delta_M(A, x) = \Delta_N(A, x); x \in L(M) \iff x \in L(N)$

Concatenation : $N = (Q = Q_1 \cup Q_2, \Sigma, \delta_N, q_1 \in N_1, F_2 \in N_2)$ $\delta_N(q, a) : \delta_1\ ((q \in Q_1 \& q \notin F_1) \| (q \in F_1 \& a \neq \varepsilon)), \delta_1 \cup \{q_1 \in N_2\}\ (q \in F_1 \& a = \varepsilon)$, $\delta_2\ (q \in Q_2)$

To DFA : state table for 2^{Q_N}, remove unreachable states, accepting $\subseteq 2^{Q_N}$ containing accepting of NFA

1.3 Non-regular languages



Pumping Lemma : A regular $\implies \exists p$ (pumping length) s.t. $\forall s \in A : |s| \geq p$ $\exists(x, y, z) : s = xyz$ s.t.; $\forall i \geq 0 : xy^iz \in A$, $|y| \geq 1, |xy| \leq p$

2 Computability

2.1 Turing Machine

Turing Maching TM : $(Q, \Sigma, \Gamma, \gamma, q_0, q_a, q_r)$, Q, Σ, Γ finite sets, $\sqcup \notin \Sigma, \Gamma$: tape alphabet $\sqcup \in \Gamma \& \Sigma \subseteq \Gamma, \delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, $q_a \in Q$ accepting state, $q_r \in Q$ reject state $q_a \neq q_r$

Representation : $uqv, u, v \in \Gamma^*, q \in Q, q$ curr. state, uv tape content, first symbol of v is head location

Transitions : conf. $uaq_ibv, a, b \in \Gamma, u, v \in \Gamma^*$, $q_i \in Q$; move to uq_jacv if $\delta(q_i, b) = (q_j, c, L), uacq_jv$ if $\delta(q_i, b)(q_j, c, R)$

Computation : start conf. $C_1 = q_0w$ for input $w \in \Sigma^*$, valid moves/transitions, accept (reject) & halt if q_a (q_r) reached

Turing-Recognizable : M recognizes $L \subseteq \Sigma^*$ $\iff \forall w \in \Sigma^* : w \in L \implies M$ accepts w & $w \notin L \implies M$ doesn't halt or rejects w

Turing-Decidable : M decides $L \subseteq \Sigma^* \iff \forall w \in \Sigma^* : M$ halts on w & M accepts $w \iff w \in L$

Halting problem : $\mathrm{HALT} = \{\langle M, w\rangle :$ M halts on $w\}$ is undecidable but recognizable, $\overline{\mathrm{HALT}}$ not recognizable

Theorem : L decidable $\iff L, \bar{L}$ recognizable

Regular problem : $\mathrm{REG_{TM}} = \{\langle N\rangle :$ $L(N)$ regular$\}$ is undecidable

2.2 Reductions

Reducibility : Use a complexe language to reason about another one

Reduction : A reduces to B, show that solving A is sufficient to solve B

Computable function : $F : \Sigma^* \to \Sigma^*$ is computable $\iff \exists$ TM s.t. halts $\forall w$ with just $f(w)$ on its tape

Mapping reducible : language A is mapping reducible to language $B : A \leq_m B \iff \exists f$ computable function s.t. $\forall w \in \Sigma^* : w \in A \iff f(w) \in B$

Theorem : $A \leq_m B$ and B decidable (recognizable) $\implies A$ decidable (recognizable); $A \leq_m B$ and A undecidable (unrecognizable) $\implies B$ undecidable (unrecognizable)

3 Efficiency

3.1 Time Complexity

Time complexity : M decider, $t : \mathbb{N} \to \mathbb{N}$, $t(n) = \max_{w \in \Sigma^* : |w| = n}$ steps M takes on w

Big-O : $f, g : \mathbb{N} \to \mathbb{R}_+, f(n) = O(g(n)) \iff \exists C > 0, n_0 \in \mathbb{N}$ s.t. $\forall n \geq n_0\ f(n) \leq C \cdot g(n)$

Small-o : $f(n) = o(g(n)) \iff \forall c > 0, \exists n_0 \in \mathbb{N}$ s.t. $\forall n \geq n_0\ f(n) < c \cdot g(n)$

Class : $\mathrm{TIME}(t(n)) = \{L \subseteq \Sigma^* \mid L$ decided in $O(t(n))\}$

P : decidable in polynomial time on deterministic TM, $\mathbf{P} = \bigcup_{k=1}^{\infty} \mathrm{TIME}(n^k)$

Verifier for language L : TM M s.t. $\forall x \in \Sigma^*$: $x \in L \implies \exists C$ s.t. M accepts $\langle x, C\rangle$, $x \notin L \implies \forall C\ M$ rejects $\langle x, C\rangle$; C certificate/witness

Nondeterministic Turing Machine NTM : $\delta : (Q \times \Gamma) \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$, several possible transitions

Nondet. decider for language L : NTM N s.t. $\forall x \in \Sigma^*$, every computation of N on x halts + : $x \in L \implies \exists$ computation N on x accepts, $x \notin L \implies \forall$ comp. N on x rejects

Polynomial Nondet. decider : its longest computation on x is polynomial in $|x|$

Theorem : L has nondet. poly-time decider $\iff L$ has a poly-time verifier

NP : class of languages that have poly-time (nondet.)verifiers : running time on any $\langle x, C\rangle$ polynomial in $|x|$, $\mathbf{P} \subseteq \mathbf{NP}$

Cook-Levin : SAT $\in P \iff P = NP$

3.2 Polynomial-Time Reductions

Poly-time computable func. : $f : \Sigma^* \to \Sigma^*, \exists$ some poly-time TM M s.t. halts $\forall w$ with just $f(w)$ on its tape

Poly-time mapping : Language A is poly-time mapping reducible to B, $A \leq_P B$, if $\exists$ poly-time comp. func. f s.t. $\forall w \in \Sigma^* : w \in A \iff f(w) \in B$

Theorem : $A \leq_P B$ and $B \in \mathrm{P} \implies A \in \mathrm{P}$

Transitivity : $A \leq_P B$ and $B \leq_P C \implies A \leq_P C$

NP-completeness : $L \in \mathrm{NP}$ and $\forall L' \in \mathrm{NP}$: $L' \leq_P L$

NP-complete proof : give poly-time verifier for L, show $\mathrm{SAT} \leq_P L$ (or any NP-complete L^*)

Clique : k-clique is subset of k pairwise connected vertices

Vertex Cover : $G = (V, E)$, vertex cover is subset S of V s.t. $\forall e \in E$ e is incident to at least one vertex in S; $S \subseteq V$ is vertex cover $\Leftrightarrow \bar{S} = (V \setminus S)$ is independent set

Independent set : subset of pairwise non-adjacent vertices

Complement : $\bar{G} = (V, \bar{E}), \bar{E}$ s.t. $uv \in \bar{E} \Leftrightarrow uv \notin E$; $S \subseteq V$ is independent set $\Leftrightarrow S$ is a clique of $\bar{G}$