CS3231 Midterm 1 Printed Notes

Chong Chin Herng

September 2024

1 Preliminaries

Alphabet An alphabet is a finite non-empty set of symbols.

String A string is a finite sequence of symbols chosen from a given alphabet. The empty string is denoted as ϵ.

Length of a string The length of a string is the number of symbols in the string.

Powers of an alphabet Σ^k is the set of all possible strings over Σ of length k. $\Sigma^{\leq k}$ is the set of all possible strings over Σ of length at most k.

Concatenation of strings If $x = x_1x_2 \cdots x_m$ and $y = y_1y_2 \cdots y_n$, then the concatenation of x and y, denoted as $x \cdot y$, is $x_1x_2 \cdots x_my_1y_2 \cdots y_n$.

Substring A string s is a substring of a string $x = x_1x_2 \cdots x_n$ if and only if there exists $1 \leq i, j \leq n$ such that $s = x_ix_{i+1} \cdots x_j$.

Subsequence A string s is a subsequence of a string $x = x_1x_2 \cdots x_n$ if and only if there exists $1 \leq i_1, i_2, \cdots, i_k \leq n$ such that $i_1 < i_2 < \cdots < i_k$ and $s = x_{i_1}x_{i_2} \cdots x_{i_k}$.

Language A language over an alphabet is a set of strings over the alphabet.

For any language L, L_1 and L_2,

- $L_1 \cdots L_2 = L_1L_2 = \{xy : x \in L_1, y \in L_2\}$
- $L^* = \{x_1x_2 \cdots x_n : x_1, x_2, \cdots, x_n \in L, n \in \mathbb{N}\} = \{\epsilon\} \cup L \cup LL \cup \cdots$
- $L^+ = \{x_1x_2 \cdots x_n : x_1, x_2, \cdots, x_n \in L, n \geq 1\} = L \cup LL \cup \cdots$

Theorem 1.1 Number of strings over any alphabet is countable.

Theorem 1.2 Number of languages over any alphabet is uncountable.

2 DFA, NFA, Regular Expressions

Deterministic Finite Automaton (DFA) A DFA $(Q, \Sigma, \delta, q_0, F)$ is a finite automaton where

- Q is a finite set of states
- Σ is a finite set of input symbols
- $\delta : Q \times \Sigma \to Q$ is a transition function
- $q_0 \in Q$ is a starting state
- $F \subseteq Q$ is a finite set of accepting states

Additionally, $\hat{\delta} : Q \times \Sigma^*$ is defined recursively as

$$\hat{\delta}(q, \epsilon) = q$$
$$\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$$

Regular language A regular language is a language accepted by some finite automaton.

Transition diagram A transition diagram is a pictorial representation of a DFA.

Transition table A transition table is a tabular representation of a DFA.

Language accepted by a DFA A language L is accepted by a DFA A if and only if $L = L(A) = \{w | \hat{\delta}(q_0, w) \in F\}$.

Dead state A state q of a DFA is a dead state if and only if for all $w \in \Sigma^*$ we have $\hat{\delta}(q, w) \notin F$.

Unreachable state A state q of a DFA is an unreachable state if and only if for all $w \in \Sigma^*$ we have $\hat{\delta}(q_0, w) \neq q$.

Nondeterministic Finite State Automaton (NFA) An NFA $(Q, \Sigma, \delta, q_0, F)$ is a finite automaton where

- Q is a finite set of states
- Σ is a finite set of input symbols
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is a transition function
- $q_0 \in Q$ is a starting state
- $F \subseteq Q$ is a finite set of accepting states

Additionally, $\hat{\delta} : Q \times \Sigma^*$ is defined recursively as

$$\hat{\delta}(q, \epsilon) = \{q\}$$
$$\hat{\delta}(q, xa) = \bigcup_{p \in \hat{\delta}(q, x)} \delta(p, a)$$

Language accepted by an NFA A language L is accepted by an NFA A if and only if $L = L(A) = \{w|\hat{\delta}(q_0, w) \cap F \neq \emptyset\}$.

Theorem 2.1 There is a DFA that accepts a given language if and only if there is an NFA that accepts the same language.

ϵ-NFA An ϵ-NFA $(Q, \Sigma, \delta, q_0, F)$ is a finite automaton where

- Q is a finite set of states
- Σ is a finite set of input symbols
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$ is a transition function
- $q_0 \in Q$ is a starting state
- $F \subseteq Q$ is a finite set of accepting states

Additionally, for any state $q \in Q$, the ϵ-closure of q, denoted as $Eclose(q)$, is a subset of Q is defined recursively as

$$q \in Eclose(q)$$
$$\text{If } p \in Eclose(q), \text{ then each state in } \delta(p, \epsilon) \text{ is in } Eclose(q)$$

and membership of $Eclose(q)$ can always be demonstrated by finitely many applications of the above two statements.

$\hat{\delta} : Q \times \Sigma^*$ is defined recursively as

$$\hat{\delta}(q, \epsilon) = Eclose(q)$$
$$\hat{\delta}(q, wa) = \bigcup_{p \in R} Eclose(p), \text{ where } R = \bigcup_{p \in \hat{\delta}(q,w)} \delta(p, a)$$

Language accepted by an ϵ-NFA A language L is accepted by an ϵ-NFA A if and only if $L = L(A) = \{w|\hat{\delta}(q_0, w) \cap F \neq \emptyset\}$.

Theorem 2.2 There is an ϵ-NFA that accepts a given language if and only if there is a DFA that accepts the same language.

Regular expression A regular expression is a method to describe a language over Σ and is defined recursively as

- ϵ and $\emptyset$ are regular expressions, and $L(\epsilon) = \{\epsilon\}$ and $L(\emptyset) = \emptyset$
- For all $a \in \Sigma$, a is a regular expression, and $L(a) = \{a\}$
- If r_1, r_2 are regular expressions, then $r_1 + r_2$, $r_1 \cdot r_2$, r_1^* and (r_1) are regular expressions
- $L(r_1 + r_2) = L(r_1) \cup L(r_2)$
- $L(r_1 \cdot r_2) = \{xy|x \in L(r_1) \text{ and } y \in L(r_2)\}$
- $L(r_1^*) = \{x_1x_2 \cdots x_k| \text{ for all } 1 \leq i \leq k, x_i \in L(r_1)\}$
- $L((r_1)) = L(r_1)$

In regular expressions, $*$ has the highest precedence, followed by $\cdot$, followed by $+$.

Theorem 2.3 There is a regular expression that can describe a language accepted by some DFA.

Theorem 2.4 There is an ϵ-NFA that can accept a language described by some regular expression.

Theorem 2.5 For any regular expression L, M and N we have

- $M + N = N + M$
- $L(M + N) = LM + LN$
- $L + L = L$
- $(L^*)^* = L^*$
- $\emptyset^* = \epsilon$
- $\epsilon^* = \epsilon$
- $L^+ = LL^* = L^*L$
- $L^* = \epsilon + L^+$
- $(L + M)^* = (L^*M^*)^*$

3 Minimization of DFA

Theorem 3.1 Let L be a regular language over Σ. Define the equivalence relation $\equiv_L$ on Σ^* such that $u \equiv_L w$ if and only if for all $x \in \Sigma^*$ we have $ux \in L \iff wx \in L$. Note that since L is regular, $\Sigma^*/\equiv_L$ must be finite. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA where

$$\begin{aligned} Q &= \{[w] : w \in \Sigma^*\} \\ q_0 &= [\epsilon] \\ F &= \{[w] : w \in L\} \\ \delta([w], a) &= [wa] \end{aligned}$$

Note that δ is well-defined as $u \equiv_L w \implies ua \equiv_L wa$. A is the unique minimal (with respect to $|Q|$) DFA for L.

Distinguishable states Two states (p, q) of a DFA are distinguishable if and only if there exists a string w over the relevant alphabet such that either

$$\hat{\delta}(p, w) \in F \text{ and } \hat{\delta}(q, w) \notin F, \text{ or}$$
$$\hat{\delta}(p, w) \notin F \text{ and } \hat{\delta}(q, w) \in F$$

Additionally, (p, q) are indistinguishable if it is not indistinguishable.

Algorithm 3.2 Given a DFA with transition function δ, the algorithm below determines all pairs of indistinguishable states.

1. Each pair of states (p, q) such that $p \in F$ and $q \notin F$ is distinguishable.
2. If $\delta(p, a)$ and $\delta(q, a)$ are distinguishable, then (p, q) is distinguishable.
3. Repeat step 2 until no new pairs of distinguishable states are discovered.
4. The remaining pairs of states are indistinguishable.

Algorithm 3.3 Given DFA $(Q, \Sigma, \delta, q_0, F)$ accepting L, the algorithm below forms the unique minimal DFA accepting L.

1. Delete all unreachable states.
2. Invoke algorithm 3.2 to find all pairs of indistinguishable states.
3. Define an equivalence relation $\sim$ on the set of states such that two states are equivalent if and only if they are indistinguishable.
4. The DFA $(Q/\sim, \Sigma, \delta_{new}, [q_0], \{[q] | q \in F\})$ where $\delta_{new}([p], a) = [q]$ if and only if $\delta(p, a) = q$ is the unique minimal DFA accepting L.

4 Properties of Regular Languages

Theorem 4.1 (Pumping Lemma) Let L be a regular language. There exists $n \in \mathbb{Z}^+$ such that for all $w \in L$ satisfying $|w| \geq n$, we can write $w = xyz$ such that

- $y \neq \epsilon$
- $|xy| \leq n$
- For all $k \in \mathbb{N}$ we have $xy^k z \in L$

Theorem 4.2 For any language L, L_1 and L_2 over Σ,

- If L_1, L_2 are regular, then $L_1 \cup L_2$ is regular.
- If L_1, L_2 are regular, then $L_1 \cdot L_2$ is regular.
- If L is regular, then $\overline{L} = \Sigma^* - L$ is regular.
- If L_1, L_2 are regular, then $L_1 \cap L_2$ is regular.
- If L_1, L_2 are regular, then $L_1 - L_2$ is regular.
- If L is regular, then $L^R = \{x^R | x \in L\}$ is regular.

5 Homomorphism

Homomorphism For any alphabet Σ and Γ, a mapping $h : \Sigma^* \to \Gamma^*$ is a homomorphism if and only if $h(\epsilon) = \epsilon$ and for all $a \in \Sigma$ and $w \in \Sigma^*$ we have $h(aw) = h(a)h(w)$.

Theorem 5.1 For any homomorphism h, if L is regular, then $h(L)$ is regular.

Tutorial 1

Question 2 There are $|\Sigma|^n$ strings of length n over Σ. If $|\Sigma| > 1$, then there are $\frac{|\Sigma|^{n+1}-1}{|\Sigma|-1}$ strings of length $\leq n$ over Σ. If $|\Sigma| = 1$, then there are $n+1$ strings of length $\leq n$ over Σ.

Question 3 For all languages $A, B_1, B_2, \cdots$, we have $A \cdot (\cup_{i=1}^{\infty} B_i) = \cup_{i=1}^{\infty}(A \cdot B_i)$ and $(A^*)^+ = (A^+)^* = A^*$.

Tutorial 2

Question 2 For any DFA $(Q, \Sigma, \delta, q_0, F)$ we have $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ for all strings x, y over Σ and all states $q \in Q$.

Tutorial 3

Question 3a If A and B are regular, then $A \cdot \overline{B}$ is regular.

Question 3f If L is regular, then $\{x : |x| = 2r \text{ for some } r \in \mathbb{N} \text{ and } x_1 x_3 \cdots x_{2r-1} \in L\}$ is regular.

Question 4 If L is regular, then $HALF(L) = \{w | (\exists u)[wu \in L \text{ and } |w| = |u|]\}$ is regular.