CS360 – Theory of Computing

May 16, 2019

All notes are typed by LaTeX

Important Notes:
· Computational problems and devices can be viewed as mathematical object
· Computation is very general

- Computer running program
- Networks of computers interacting
- People performing computation by hand
- Proofs of theorem
- Biological process

Mathematical fundation:
· Set theory

$$\begin{cases} Naive\ theory \\ Axiomatic\ theory \end{cases}$$

Some notation:
1. TFAE = **T**he **F**ollowing **A**re **E**quivalent

Lecture 1:

Defn: A set is countable if
· A is empty
· There exist an onto function f of this form:

$$f: \mathbb{N} \to A$$

TFAE:
1. A is countable.
2. There exist a one-to-one function g of the form:

$$g: A \to \mathbb{N}$$

3. Either A is finite or there exist a one-to-one onto function h of the form:

$$h: \mathbb{N} \to A$$

Terminology:

If A is a set, then the power set $P(A)$ is the set of all subsets of A.

Cantor Theorm:

The power set of natural number is not countable.

$$\textit{i.e. } P(\mathbb{N}) \textit{ is not countable.}$$

Prove by contradiction:
Assume towards contradiction that $P(N)$ is countable.
This implies that there exist an onto function f, such that

$$f: \mathbb{N} \to P(N)$$

Consider we define a set $S = \{n \in \mathbb{N} : n \notin f(n)\}$
Since f is onto function, there must exist some number $n \in N$, such that $S = f(n)$
We fix such a choice of n! (i.e. $S = f(n)$)
We get

$$n \in S \Leftrightarrow n \notin f(n) \Leftrightarrow n \notin S$$

Thus it is a contradiction. Therefore $P(\mathbb{N})$ is not countable.

Lecture 2:

Terminologies:
Alphabet – symbols for using computation

An alphabet is a finite and nonempty set.

Typical names: Σ, Γ
Examples:

$$\Sigma = \{0, 1\} \rightarrow \textit{binary alphabet}$$
$$\Sigma = \{0\} \rightarrow \textit{unary alphabet}$$
$$\Sigma = \{0, 1, 2, \cdots, n-1\} \textit{ for some possible interger } n$$

String – over some alphabet

Let Σ be an alphabet, the string over Σ is a finite sequence of symbols from Σ.

Typical variables names for symbols: a, b, c, d
Typical string names: u, v, w, x, y, z
Examples:

$$\textit{under } \Sigma = \{0, 1\} \textit{ we have } 0101,\ 0,\ 1,\ \epsilon$$
$$\textit{under } \Sigma = \{0, 1, 2\} \textit{ we have } 01,\ 1,\ 0$$

Language

Let Σ be an alphabet. A language over Σ is a set of strings over Σ. Σ^* language consisting of EVERY string over Σ.

Typical names: A, B, C, D
Example(assume $\Sigma = \{0, 1\}$):

$$B = \{w \in \Sigma^* : w \textit{ ends with } 0\}$$
$$C = \{w \in \Sigma^* : |w| \textit{ is a prime number}\}$$

Question:

Consider Σ^*. Is it countable?
Yes, it is!
Consider we can find an onto function f

$$f : \mathbb{N} \to \Sigma^*$$

We arrange the all the strings in Σ^* by its length.
i.e. ϵ, 0, 1, 00, 01, 10, 11, $\cdots$
The sequence above has lexicographic order
Then we set the function to be:

$$f(0) = \epsilon$$
$$f(1) = 0$$
$$f(2) = 1$$
$$f(3) = 00$$
$$\cdots$$

Thus we can form an onto function that is able to count all the strings.
Therefore Σ^* is countable.

Consider $\Sigma = \{0, 1\}$, how many languages over Σ are there?
TFAE:
1. A is language over Σ
2. $A \subset \Sigma^*$
3. $A \in P(\Sigma^*)$

The answer is No. We can prove this by using Cantor Theorem.
From the previous questions, we know that there is an one-to-one function f that can map from Σ^* to $\mathbb{N}$.

$$i.e.\ \ f : \Sigma^* \to \mathbb{N}$$

Assume towards contradiction that $P(\Sigma^*)$ is countable.
Therefore we can find an onto function, say g, that

$$g : \mathbb{N} \to P(\Sigma^*)$$

Since we can map $\mathbb{N}$ to Σ^*, thus we can find an onto function, say h, such that

$$h : \mathbb{N} \to P(\mathbb{N})$$

That contradicts Cantor's Theorem.
Therefore we get the languages over Σ is uncountable.

Deterministic Finite Automata (DFA)

Definition:
A DFA is a 5 tuple $m = (Q, \Sigma, \delta, q_0, F)$ where:

- Q is a finite nonempty set of states.
- Σ is an alphabet.
- δ is the transition function of the form

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

- q_0 is the starting state.
- F is the set of accepting states.

Important notes:
A DFA needs to accept all the strings in one language AND rejects all the strings that are not in that language.

Formal Definition of DFA
Let M be a DFA and let $w \in \Sigma^*$. The DFA m accepts w if one of these following holds:
1. $\omega = \epsilon$ and $q_0 \in F$
2. $\omega = a_1 a_2 \cdots a_n$ for $n \leq 1$ and $a_1, a_2, \cdots, a_n \in \Sigma$ and there exist a sequence of states $r_0, r_1, \cdots, r_n \in Q$ such that $r_0 = q_0$, $r_n \in F$ and $r_{k+1} = \delta(r_k, a_{k+1})$ for every $k \in \{0, 1, 2, \cdots, n-1\}$. If m does not accept, then it rejects w.

Convention Function:
Suppose $m = (Q, \Sigma, \delta, q_0, F)$ is a DFA.
Define $\delta^* : Q \times \Sigma^* \to Q$ as follows:

- $\delta^*(q, \epsilon) = q$
- $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$

Notation:
$L(M) = \{w \in \Sigma^* : M \text{ accepts } w\} \rightarrow$ "language recognized by M"

Regular Language Definition:
Let Σ be an alphabet and let A be a language over Σ. A is regular if there exist a DFA such that $L(M) = A$

Consider fixed $\Sigma = \{0, 1\}$, how many regular language over Σ?
The answer is countable many.
Consider the number of states.
Similar to the approach above.
We can list all the DFA by counting their states.
Then we can claim that there exist an onto function f that maps from $\mathbb{N}$ to all the DFA over Σ.
Thus we know that DFA is countable.
Therefore by the definition above, we know regular language is countable.

Lecture 3:

Non-Deterministic finite automatas(NFAs)

— verification whether computer can get the result by trying multiple ways.

Defn:
NFA is a 5-tuple:

$$N = (Q, \Sigma, \delta, q, F)$$

where

- Q is a finite and non-empty set of states
- Σ is a transition function of the form

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow P(Q)$$

- $q_0 \in Q$ is a start state
- $F \subset Q$ is a set of accept states

Formal Definition
Let $N = \{Q, \Sigma, \delta, q_0, F\}$ be an NFA and let $w \in \Sigma^*$. N accepts w if there exist $m \in \mathbb{N}$, a sequence of states, $r_0 \cdots r_m \in Q$ and a sequence of alphabet (move), $a_1, \cdots, a_m \in \Sigma \cup \{\epsilon\}$

- $r_0 = q_0$, $r_m \in F$
- $r_{k+1} \in \delta(r_k, a_{k+1}),\ \forall\ k \in \{0, 1, \cdots, m-1\}$
- $w = a_1 a_2 \cdots a_m$ If N does not accept w, then rejects w

Now we need to check the base case when $m = 0$.
Consider $m = 0$, then we have

$$\begin{cases} r_0 \cdots r_m \to r_0 \in F \\ a_1 \cdots a_0 \to \textit{empty sequence} \end{cases}$$

From above, we get the starting state is in the set of accepting states.
Therefore, we know the condition satisfy all situations.

ϵ - closure
Let $R \subset Q$ and let $N = \{Q, \Sigma, \delta, q_0, F\}$ be an NFA. The ϵ – closure of R denoted $\epsilon(R)$ is the set of all states that can be reached from any states in R by following 0 or more ϵ transition.

$$\epsilon : P(Q) \to P(Q)$$

Define δ^* for an NFA

$$\delta^*(p, \epsilon) = \epsilon(\{p\})$$
$$\delta^*(p, wa) = \epsilon(\cup_{r \in \epsilon^*(p,w)} \epsilon(r, a))$$

Theorem:
Let Σ be an alphabet, A $\subset \Sigma^*$ be a language.
The language A is regular iff there exist an NFA, N $= \{Q, \Sigma, \Delta, q_0, F\}$ such that $A = L(N)$
proof:
$\Rightarrow$ given that A is regular, then there exist a DFA $M = (Q, \Sigma, \Delta, q_0, F)$ with A = L(N)
Define $N = (Q, \Sigma, \gamma, q_0, F)$ where

$$\gamma(q, a) = \{\delta q, a\}$$
$$\gamma(q, \Sigma) = \emptyset \ \forall \ q \in Q, \ a \in \Sigma$$

In this case, L(N)=L(M)=A.

$\Leftarrow$ given that NFA, $N = (Q, \Sigma, \delta, q_0, F)$
idea: we will construct a DFA m, such that L(m) = L(N). We will take the state set of m to be P(Q).
define m as follows:

$$m = (P(Q), \Sigma, \gamma, \epsilon(\{q\}), G)$$

where γ and G are as follows.
$\gamma : P(Q) \times \Sigma \to P(Q)$ where $\gamma(R, a) = \epsilon(\cup_{r \in R} \delta(r, a))$ for $R \in Q, a \in \Sigma$
$G = \{R \in P(Q) : R \cap F \neq \emptyset\}$

Lecture 4:

Regular Operation:
Let $A, B \subset \Sigma^*$ be language over a alphabet Σ
The regular operation is:
1. Union: $A \cup B = \{w : w \in A \ or w \in B\}$
2. Concatenation: $AB = \{wx : w \in A \ and \ x \in B\}$
3. Kleene star: $A^* = \{\epsilon\} \cup A \cup AA \cup AAA \cdots$

Theorem:
Let Σ be an alphabet and let $A, B \subset \Sigma^*$ be regular language. The language $A \cup B, AB, A^*$ are regular.
Proof:
Consider let $M_a = (P, \Sigma, \delta, p_0, F)$ and let $M_b = (Q, \Sigma, \gamma, q_0, G)$ be DFAs with $A = L(M_a), B = L(M_b)$. WLOG, let $P \cap Q = \emptyset$
Consider $A \cup B$, we define a NFA as follows:

$$N = (P \cup Q \cup \{r_0\}, \Sigma, \mu, r_0, F \cup G)$$

where,

$$\begin{aligned}
\mu(p, a) &= \{\delta(p, a)\} \quad for \ p \in P, \ a \in \Sigma \\
\mu(p, \epsilon) &= \emptyset \quad for \ p \in P, \ a \in \Sigma \\
\mu(q, a) &= \{\gamma(q, a)\} \quad for \ q \in Q, \ a \in \Sigma \\
\mu(q, \epsilon) &= \emptyset \quad for \ q \in Q, \ a \in \Sigma \\
\mu(r_0, \epsilon) &= \{q_0, p_0\} \quad for \ a \in \Sigma \\
\mu(r_0, a) &= \emptyset \quad for \ a \in \Sigma
\end{aligned}$$

The proof for AB, A^* is similar.

Questions:
$A \subset \Sigma^*$, define $\bar{A} = \Sigma^* \setminus A = \{x \in \Sigma^* : x \notin A\}$
If A is regular is $\bar{A}$ regular?
Yes, by swaping all accepting state with non-accepting states.
Then we can form a new DFA/NFA, so we know $\bar{A}$ is regular.

If $A, B \subset \Sigma^*$ are regular is $A \cap B$ regular?
$m = (P \times Q, \Sigma, \mu, (p_0, q_0), F \times G)$
$\mu((p, q), a) = (\delta(p, a), \gamma(q, a))$

Another way is using DeMorgan's Laws.
$A \cap B = \overline{\bar{A} \cup \bar{B}}$

Regular Expression:

Defn: Let Σ be an alphabet. R is a regular expression over Σ if one of these following is true:
1. $R = \emptyset$
2. $R = \epsilon$
3. $R = a \ for \ a \ \in \Sigma$
4. $R = (R_1, R_2)$ for $a \in \Sigma$
5. $R = (R_1 R_2)$ for R_1, R_2 regular expressions
6. $R = (R_1^*)$ for R_1 regular expressions

Let R be regular expression over Σ. The language recognized or method by R is defined as follows:
1. If $R = \emptyset$, then $L(R) = \emptyset$
2. If $R = \epsilon$, then $L(R) = \{\epsilon\}$
3. If $R = a$ for $a \in \Sigma$, then $L(R) = \{a\}$
4. If $R = (R_1 \cup R_2)$, then $L(R) = L(R_1) \cup L(R_2)$
5. If $R = (R_1 R_2)$, then $L(R) = L(R_1)L(R_2)$
6. If $R = (R_1^*)$, then $L(R) = L(R_1)^*$

Order of precedure:
1. Kleene star
2. Concatenation
3. Union

Theorem: Let Σ be an alphabet and let $A \subset \Sigma^*$ be a language over Σ. A is regular if and only if there exist a regular expression R with $L(R) = A$.