Theorey of Computation Notes

Yuxuan Sun

Spring 2022

Contents

1 regular languages **2**

2 Non-regular languages **2**
2.1 Exercises using pumping lemma 3

3 Context-Free Grammars **6**

4 Push-Down Autonoma **8**

5 decidability **11**
5.1 undecidability 12

6 reducibility **14**
6.1 exercises 15

1 regular languages

Definition 1.1: regular language

language is called a regular language if some finite automaton recognizes it.

2 Non-regular languages

Theorem 2.1: Pumping Lemma

If L is a regular language, then there is a positive integer n (typically, n is the number of states of the DFA accepting L) such that, if $x \in L$ and $|x| \geq n$, then there exist $u, v, w \in \Sigma^*$ such that $x = uvw$ and:

1. $|uv| \leq n$
2. $|u| > 0$
3. for each integer $m \geq 0$, $uv^m w \in L$

Corollary 2.2: infinite

Let the regular language L be accepted by a DFA with n states. Then L is infinite if and only if there is $x \in L$ s.t. $n \leq |x| < 2n$.

Theorem 2.3: Complement

The class of regular languages is closed under complement.

Definition 2.4: Indistinguishable Strings

Let L a language over Σ and let $x, y \in \Sigma^*$. We say that x and y are indistinguishable with respect to L and we write $x \approx_L y$ if, for all $z \in \Sigma^*$, either both xz and $yz \in L$ or neither is. Furthermore, $\approx_L$ can be proved to be an equivalence relation on Σ^*.

Theorem 2.5: Myhill-Nerode

A language L is regular if and only if the number of equivalence classes of $\approx_L$ is finite.

2.1 Exercises using pumping lemma

The followings are proofs of L NOT being a regular language using the pumping lemma.

$L = \{x \in (a+b)^* | x \neq x^R\}$

Prove by contradiction.

Suppose L is regular. Then there exists $n \in \mathbb{N}$ s.t. let $x = a^n b a^{n+n!}$. Because $n \neq n + n!$, $x \in L$. Because $|x| = n + 1 + n + n! > n$, there exists $u, v, w \in \Sigma^*$ s.t. $x = uvw$

Based on the pumping lemma, we know that $|uv| \leq n$, based on the construction of x, v could only contain a's, denote as $v = a^i$ for some $i \in \mathbb{Z}_{>0}$. Because i is an interger that's smaller than n, we know that $n!/i$ will be an integer, denoted as m. i.e. $im = n!$. For later reference, denote $u = a^j, w = a^k b a^{n+n!}$, where $j, k \in \mathbb{Z}_{\geq 0}$, and $j + i + k = n$

Notice that $uv^{m+1}w = uvv^m w = a^{j+i+mi+k} b a^{n+n!}$, since we know $j+i+k = n$ and $mi = n!$, we have $uv^{m+1}w = a^{n+n!} b a^{n+n!}$, which is not in L. We have a contradiction to the pumping lemma, thus L is not regular.

$L = \left\{a^i b^j \mid \frac{1}{2}(j+1) \leq i \leq \frac{1}{2}(3j-1), i, j \in \mathbb{Z}, j \geq 0\right\}$

The language could be rewritten as $L = \left\{a^i b^j \mid (j+1) \leq 2i \leq (3j-1), i, j \in \mathbb{Z}, j \geq 0\right\}$ so that it's easier to check whether our string is in the language.

Prove by contradiction.

Suppose L is regualr. Then there exists $n \in \mathbb{N}$ s.t. let $x = a^n b^n$, the inequality holds as long as $n \geq 1$ so $x \in L$. Beacuse $|x| = 2n > n$, there exists $u, v, w \in \Sigma^*$ s.t. $x = uvw$

Based on the pumping lemma, we know that $|uv| \leq n$, based on the construction of x, u, v could only contain a's. Denote $u = a^i, v = a^j$ where $i, j \in \mathbb{Z}_{\geq 0}$ but $j \neq 0$. Denote $w = a^k b^n$, $k \in \mathbb{Z}_{\geq 0}$.

Find any m s.t. $mj > 3n$, since $j \neq 0$ we know we could find such m. Then notice that $uv^m w = a^{i+mj+k} b^n$. It is not in L since $2(i + mj + k) > 3n > 3n - 1$, which violates the inequality of language. We have a contradiction to the pumping lemma, thus L is not regular.

$L = \{x \in (a+b)^* \mid n_a(x) \neq 10 n_b(x)\}$

We need to prove by its complement, namely

$$\overline{L} = \{x \in (a+b)^* | n_a(x) = 10 n_b(x)\}$$

Prove by contradiction.

Suppose $\overline{L}$ is regular. Then there exists $n \in \mathbb{N}$ s.t. $x = a^{10n}b^n$. Because $10n = 10n$, $x \in \overline{L}$. Because $|x| = 10n + n > n$, there exists $u, v, w \in \Sigma^*$ s.t. $x = uvw$.

Based on the pumping lemma, we know that $|uv| \leq n$, based on the construction of x, u, v could only contain a's. Denote $u = a^i, v = a^j, w = a^k b^n$ where $i, j, k \in \mathbb{Z}_{\geq 0}$ but $j \neq 0$, and $i + j + k = 10n$.

Take any $m \geq 1$, then $uv^m w = a^{i+mj+k}b^n$. Because $j \neq 0$ and $m \neq 0$, we know that $i + mj + k > i + j + k$, i.e. $i + mj + k > 10n$, so the inequality is broken as $i + mj + k \neq 10n$. We have a contradiction to the pumping lemma, thus $\overline{L}$ is not regular, and according to theorem in class, L is not regular.

$$L = \left\{a^i b^j c^k \mid j = |i - k|, i, j, k \in \mathbb{Z}, i, k \geq 0\right\}$$

Prove by contradiction.

Suppose L is regular. Then there exists $n \in \mathbb{N}$ s.t. $x = a^{2n}b^n c^n$. Because $n = |2n - n|$, $x \in L$. Because $|n| = 2n + n + n = 4n > n$, there exists $u, v, w \in \Sigma^*$ s.t. $x = uvw$.

Based on the pumping lemma, we know that $|uv| \leq n$, based on the construction of x, u, v could only contain a's. Denote $u = a^i, v = a^j, w = a^k b^n c^n$ where $i, j, k \in \mathbb{Z}_{\geq 0}$ but $j \neq 0$, and $i + j + k = 2n$.

Take any $m \geq 1$, then $uv^m w = a^{i+mj+k}b^n c^n$. Because $m \neq 0$ and $j \neq 0$, $i+mj+k > 2n$. In other words, $|i+mj+k-n| > |2n-n|$, so $|i+mj+k-n| \neq n$. $uv^m w \notin L$. We have a contradiction to the pumping lemma, thus L is not regular.

$$L = \left\{a^i b^j c^k \mid i, j, k \in \mathbb{Z}, i, j, k \geq 0 \text{ such that if } i = 1, \text{ then } j = k\right\}$$

We need to prove by its complement, namely

$$\overline{L} = \left\{a^i b^j c^k \mid i, j, k \in \mathbb{Z}, i, j, k \geq 0 \text{ s.t. } i = 1 \text{ and } j \neq k\right\}$$

Prove by contradiction.

Suppose $\overline{L}$ is regular. Then there exists $n \in \mathbb{N}$ s.t. $x = ab^n c^{n+n!}$. Becuase $n \neq n + n!$ and $i = 1$, $x \in \overline{L}$. Because $|x| = 1 + n + n + n \mapsto n$, there exists $u, v, w \in \Sigma^*$ s.t. $x = uvw$.

Based on the pumping lemma, we know that $|uv| \leq n$, based on the construction of x, there are several possible situations of v, specifically:

1. $v = a$

   This will give us a contradiction because $uv^2 w = a^2 w$, but in $\overline{L}$ we don't allow i(the power of a) to be anything but 1.

2. $v = ab^i$ for some $i \in \mathbb{Z}_{>0}$

   This will give us a contradiction because $uv^2w = ab^iab^iw$, which is clearly not in $\overline{L}$ because the language doesn't allow sth like aba.

3. $v = b^j$ for some $j \in \mathbb{Z}_{>0}$

   The contradiction is similar with the one in **1.4.1**. Denote $u = ab^i, w = b^kc^{n+n!}$ where $i, k \in \mathbb{Z}_{\geq 0}$, and $i + j + k = n$. Since $j < n$, $n!/j$ will give us an integar, let's denote it as m, i.e. $mj = n!$.

   Notice $uv^{m+1}w = ab^{i+j+k+mj}c^{n+n!}$, based on what we know, $i + j + k + mj = n + n!$, i.e. $uv^{m+1}w \notin \overline{L}$. We have a contradiction to the pumping lemma, based on the theorem in class, we know $\overline{L}$ not regular implies L is not regular either.

That all possible situations for v because $|uv| \leq n$ so there is no way v could contain c.

3 Context-Free Grammars

Definition 3.1: Context-Free Grammars

A **context-free** grammar is a 4-tuple $G = (V, \Sigma, S, P)$s.t.:

1. V is a finite set of variables, $S \in V$ is the start variable
2. Σ is a finite set of terminal symbols or teminals s.t. $V \cap \Sigma = \varnothing$
3. P is a finite seet, whose elements are **grammar rules** or **productions** in the form
$$A \to \alpha$$
where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$

Definition 3.2: The Language Generated by a CFG

If $G = (V, \Sigma, S, P)$ is a CFG, the language generated by G is

$$L(G) = \{x \in \Sigma^* \mid S \Longrightarrow_G^* x\}$$

Language L is a context-free language if there is a CFG G s.t. $L = L(G)$

Definition 3.3: derivation tree/parse tree

Let G be a CFG. The **derivation tree** for G is an ordered tree s.t.

1. the root is labeled S
2. every leaf has a label from $\Sigma \cup \{\epsilon\}$
3. every interior vertex has a label from V
4. if a vertex has label A, and its children are labeled (left to right) $a_1, a_2, \ldots, a_n$, where $a_j \in V \cup \Sigma \cup \{\epsilon\}$ for $j = 1, 2, 3, \ldots, n$, then P contains a production of the form $P \to a_1 a_2 \ldots a_n$

Definition 3.4: yield of a tree

the string of terminals obtained by reading the leaves of the tree from left to right, omitting any ϵ.

Theorem 3.5: derivation tree and yield

If G is a CFG, then, for every $x \in L(G)$, there exists a derivation tree of G whose yield is x . Conversely, the yield of any derivation tree is in $L(G)$

Definition 3.6: leftmost derivation

A derivation in a CFG is a leftmost derivation if, at each step, a production is applied to the leftmost variable-occurrence in the current string.

Theorem 3.7: equivalent statement of $x \in L(G)$

1. x has more than one derivation tree
2. x has more than one leftmost derivation
3. x has more than one rightmost derivation

Definition 3.8: ambiguous CFG

A CFG G is ambiguous if, there exists $x \in L(G)$ s.t. x has more than one derivation tree.

4 Push-Down Autonoma

Definition 4.1: PDA

$M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$ where

- Q is a finite set of states
- Σ is a finite set which is called the input alphabet
- Γ is a finite set which is called the stack alphabet
- δ is a finite subset of $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$, the transition relation
- $q_0 \in Q$ is the start state
- $Z \in \Gamma$ is the initial stack symbol
- $F \subseteq Q$ is the set of accepting states

Definition 4.2: Configurations and Moves

Let $M = (Q, \Sigma, \Gamma, q_0, Z_0, F, \delta)$ be a PDA
A **configuration/instananeous discription** of M is a triplet (p, w, γ) s.t. when M is on state $p \in Q$, the part of the input string that is about to be read is string $w \in \Sigma^*$ and, the contents of the whole stack are given my string $\gamma \in \Gamma^*$
Two configurations $(p, \sigma w, Z\alpha)$ and $(q, w, \gamma\alpha)$ (for $p, q \in Q, \sigma \in \Sigma_\varepsilon, w \in \Sigma^*, Z \in \Gamma_\varepsilon, \alpha, \gamma \in \Gamma^*$) are said to form a move in one step, written as

$$(p, \sigma w, Z\alpha) \vdash (q, w, \gamma\alpha)$$

whenenver $\delta(p, \sigma, Z) \ni (q, \gamma)$

Definition 4.3: chains of moves

Let $C_0, C_1, \ldots, C_n$ be a sequence of configurations, **a chain of moves in** n **steps**, $C_0 \vdash C_1 \vdash \cdots \vdash C_n$, could be written as

$$C_0 \vdash^n C_n$$

Definition 4.4: Right-, Left-, Regular and Linear Grammars

Let $G = (V, \Sigma, S, P)$ a CFG.
G is called **left-linear** if all productions are of one of the two forms,

$$A \to xB$$
$$A \to x$$

where $A, B \in V$ and $x \in \Sigma^*$
(similar for **left-linear**)
G is called regular grammar if ti's either right- or left-linear
G is called linear grammar if at most one variable can occur on the right side of any production, independently of its position.

Definition 4.5: Chomsky Normal Form

A CFG $G = (V, \Sigma, S, P)$ is in **Chomsky normal form** if all productions are of the form

$$A \to BC$$

or

$$A \to a$$

where $A, B, C \in V$ and $a \in \Sigma$ (notice $a \neq \epsilon$)

Theorem 4.6

Modify G's productions, an equivalent CFG $\hat{G}$ in Chomsky normal form can be created.

Theorem 4.7

For every CFG G, there is a PDA M s.t. $L(M) = L(G)$

Theorem 4.8: Pumping Lemma for Context-Free Languages

If L is a context-free language over alphabet Σ, then there is a positive integer n s.t., for every $x \in L$ with $|x| \geq n$, x can be written as $x = uvwxy$ for some string $u, v, w, x, y \in \Sigma^*$ satisfying:
$|vwx| \leq n$
$|vx| \geq 1$, i.e. $v \neq \epsilon$ or $x \neq \epsilon$
for every integer $m \geq 0$, $uv^m wx^m y \in L$

Corollary 4.9

Let L be a CFL and n the positive integer from the pumping lemma, then:
$L \neq \varnothing$ iff there exsits $w \in L$ with $|w| < n$
L is infinite iff there exists $z \in L$ s.t. $n \leq |z| < 2n$

Theorem 4.10: Ogden's Lemma

If L is a context-free language over alphabet Σ, then there is a opstivie integer n s.t. for every $x \in L$ with $|x| \geq n$, if we mark at least n symbols of x, x can be written as $x = uvwxy$, for some strings $u, v, w, x, y \in \Sigma*$ satisfying
the string vwx contains at most n marked symbols
the string vx contains at least one marked symbol
for every integer $m \geq 0$, $uv^m wx^m y \in L$

Theorem 4.11

The class of CFLs is closed under union, concatenation, and Kline star
the class of CFLs is not closed under intersection and complementation
the intersection of a CFL with a regular language is a CFL

5 decidability

Definition 5.1: Turing-decidable, decidable

A language is **Turing-decidable** or **decidable** if some Turing machine decides it: always make a decision to accept of reject.

Theorem 5.2: A_{DFA} is a decidable language

Let

$$A_{DFA} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}.$$

Proof Idea

We need to present a TM M that decides A_{DFA}.

Let M be a turning machine such that:

On input $\langle B, w\rangle$, where B is a DFA and w is a string:

1. simulate B on input w
2. if the simulation ends in an accept state, *accepts. If it ends in a non-accepting state, reject.*

Theorem 5.3: A_{NFA} is a decidable language

$$A_{NFA} := \{\langle B, w\rangle \mid B \text{ is an NFA that accepts input string } w\}$$

Theorem 5.4: A_{REX} is a decidable language

$$A_{REX} := \{\langle R, w\rangle \mid R \text{ is a regular expression that generates string } w\}$$

Theorem 5.5: E_{DFA} is a decidable language

$$E_{DFA} := \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \varnothing\}$$

In other words, A doesn't accept anything (not even the empty string).

Proof

Design a turning machine T s.t. given input $\langle A\rangle$ where A is a DFA:

1. mark the start state of A
2. repeeat until no new states get marked
3. mark any state that has a transition coming into it from any state that is already market
4. if no accept state is marked, accept; otherwise reject.

Theorem 5.6: EQ_{DFA} is a decidable language.

$$EQ_{DFA} := \{\langle A, B\rangle \mid A, B \text{ are DFAs and } L(A) = L(B)\}$$

Proof

Let's use Theorem 5.5. To use it, we need to construct a DFA, denoted as C, s.t. $L(C) = \varnothing$.

To do so, we use **symmetric difference** of $L(A)$ and $L(B)$, namely

$$L(C) = \Big(L(A) \cap \overline{L(B)}\Big) \cup \Big(\overline{L(A)} \cap L(B)\Big)$$

Notice that if $L(A) \subseteq L(B)$, then $L(A) \cap \overline{L(B)} = \varnothing$, similar for the second part. Thus, if $L(A) = L(B) \iff L(C) = \varnothing$. Now we have our $L(C)$, the turning machine construction is easy.

Construct a turning machine F s.t. given input $\langle A, B\rangle$, where A, B are DFAs

1. construct DFA C as shown above
2. run a turning machine T on C as in Theorem **??** on input $\langle C\rangle$.
3. if T accepts, accept. If T rejects, reject.

Theorem 5.7: A_{CFG} is a decidable language

$$A_{CFG} := \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$$

Theorem 5.8: E_{CFG} is a decidable language

$$E_{CFG} := \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \varnothing\}$$

5.1 undecidability

Theorem 5.9: A_{TM} is undecidable

$$A_{TM} := \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$$

Notice that A_{TM} is turning-recognizable, i.e. we could build a turning machine that might loop when M loops on w. This tells us that turing-recognizable is more powerful than decidable.

Definition 5.10: co-Turing-recognizable

A language is co-Turing-recognizable if it is the complement of a Turing-reconizable language. (its complement is Turing-recognizable).

Theorem 5.11

A language is decidable iff it is Turing-recognizable and co-Turing-recognizable.

in other words, A langugae is decidable iff it is Turing-recognizable and its complement is Turing-recognizable.

in terms of undecidability, we just take the negation of the statement, namely: a language is undecidable iff it is not Turing-recognizable or its complement is not Turing-recognizable.

Corollary 5.12: $\overline{A_{TM}}$ is not Turing-recognizable

Prove by contradiction. If $\overline{A_{TM}}$ is Turing-recognizable, then by the theorem above, A_{TM} should be decidable, yet we know it's not. Thus $\overline{A_{TM}}$ has to be NOT turing-recognizable.

Theorem 5.13: E_{TM} is undecidable

$$E_{TM} := \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \varnothing\}$$

Theorem 5.14: $REGULAR_{TM}$ is not decidable

$$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language } \}$$

Theorem 5.15: EQ_{TM} is not decidable

$$EQ_{TM} := \{\langle M_1, M_2 \rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

6 reducibility

Definition 6.1: computable function

A function $f : \Sigma^* \to \Sigma^*$ is a **computable function** if some Turing machine M, on every input w, halts with just $f(w)$ on its tape.

Definition 6.2: mapping reducible

Language A is **mapping reducible** to language B, denoted as $A \leq_m B$, if there is a computable function $f : \Sigma^* \to \Sigma^*$, where for every w,

$$w \in A \iff f(w) \in B.$$

The function f is called the **reduction** from A to B.

Theorem 6.3

If $A \leq_m B$ and B is decidable, then A is decidable.

Notice the contrapositive is:

Corollary 6.4

If $A \leq_m B$ and A is undecidable, then B is undecidable.

Theorem 6.5

If $A \leq_m B$ and B is turing-recognizable, then A is turing-recognizable.

similarly, for its contrapostivie:

Corollary 6.6

If $A \leq_m B$ and A is not Turing-recognizable, then B is not Turing-recognizable.

6.1 exercises

If $A \leq_m B$ and B is a regular language, does that imply that A is a regular language?

Answer: No Recall from corollary **5.23** from book that: if $A \leq_m B$ and A is undecidable, then B is undecidable.

Take the contra-positive of the corollary, we'll have: if B is decidable then either $A \nleq_m B$ or A is decidable.

Since we have B, as a regular language, is decidable and $A \leq B$, we know that A is decidable. Thus let's find any decidable language that's not a regular expression. Take

$$A = \{a^n b^n\}$$

we know it's decidable becuase it could be regonized by a PDA. To show $A \leq_m B$, we could have the following computable function:

$$f(w) = \begin{cases} 0 & \text{if } w = a^n b^n \\ 1 & \text{otherwise} \end{cases}$$

clearly that $w \in A \iff f(w) \in B$. Thus we've found a counter example.