NOTES ON THE THEORY OF COMPUTATION

YANNAN MAO

22nd September 2022

Contents

Automata and Formal Languages

An **alphabet** is a finite set Σ, and a **word over the alphabet** Σ is a finite sequence of the elements of Σ. If a word w is the sequence $(w_0, \dots, w_n)$ for some $n \in \mathbb{N}$, we may write the word as the concatenation $w_0 \cdots w_n$. The empty word is denoted by ϵ. The set of all words over Σ is Σ^*[1]. A **formal language over the alphabet** Σ is a subset of Σ^*.

An **automaton** is an ordered sequence that **accepts** some words over an alphabet. The set of words an automaton accepts forms a language, which is unique, in which case we say the automaton **recognises** the language. Given an automaton M, we may speak of the unique language recognised by M as the **language of the automaton** M. An automaton may accept no word, in which case the language thereof is $\emptyset$.

1.1 Finite-State Automata and Regular Languages

1.1.1 Deterministic Finite-State Automata

Definition 1. A **deterministic finite-state automaton** is an ordered quintuple $(\Sigma, S, \delta, s_0, F)$ wherein

(a) Σ is an alphabet,

(b) S is a finite set of **states**,

(c) $\delta : S \times \Sigma \to S$ is the **transition function**,

(d) $s_0 \in S$ is the **initial state**, and

(e) $F \subseteq S$ is the set of **accepting states**.

Let $M = (\Sigma, S, \delta, s_0, F)$ be a deterministic finite-state automaton and let $w = w_0 \cdots w_n$ wherein $n \in \mathbb{N}$ be a word over Σ. Then M accepts w if there exists a sequence of states $(r_0, \dots, r_{n+1})$ in S such that

(a) $r_0 = s_0$,

(b) $\delta(r_i, w_i) = r_{i+1}$ for $i \in \mathbb{N}_{<n+1}$, and

(c) $r_{n+1} \in F$.

[1] * denotes the unary operator of Kleene star, defined as $A^* = \{a_0 \cdots a_n : n \in \mathbb{N} \land \forall i \in \mathbb{N}_{<n+1} (a_i \in A)\} \cup \{\epsilon\}$.

Furthermore, M accepts ϵ if $s_0 \in F$.

1.1.2 Nondeterministic Finite-State Automata

Definition 2. A **nondeterministic finite-state automaton** is an ordered quintuple $(\Sigma, S, \delta, s_0, F)$ wherein

(a) Σ is an alphabet,

(b) S is a finite set of states,

(c) $\delta : S \times (\Sigma \cup \{\epsilon\}) \to \mathscr{P}(S)$ is the transition function,

(d) $s_0 \in S$ is the initial state, and

(e) $F \subseteq S$ is the set of accepting states.

Let $M = (\Sigma, S, \delta, s_0, F)$ be a nondeterministic finite-state automaton and let w be a word over Σ. Then M accepts w if $w = w_0 \cdots w_n$ wherein $n \in \mathbb{N}$ such that each $w_i \in \Sigma \cup \{\epsilon\}$ for $i \in \mathbb{N}_{<n+1}$ and that there exists a sequence of states $(r_0, \dots, r_{n+1})$ in S such that

(a) $r_0 = s_0$,

(b) $r_{i+1} \in \delta(r_i, w_i)$ for $i \in \mathbb{N}_{<n+1}$, and

(c) $r_{n+1} \in F$.

We say that two automata are equivalent if they recognise the same language.

Theorem 1. *Every nondeterministic finite-state automaton has an equivalent deterministic finite-state automaton.*

Proof. Let $N = (\Sigma, S, \delta, s_0, F)$ be the nondeterministic finite-state automaton recognising some language A over Σ. We construct a deterministic finite-state automaton $M = (\Sigma, S', \delta', s_0', F')$ recognising A.

We first see that $S' = \mathscr{P}(S)$ and that $F' = \{R \in S' : R \cap F \neq \varnothing\}$.

Let $\delta_0 : S \times \{\epsilon\} \to \mathscr{P}(S)$ be defined as $\delta_0(s, \epsilon) = \delta(s, \epsilon)$ for each $s \in S$. Assume first that, thus induced, $\delta_0 = \varnothing$ for N. For each $R \in S'$ and for each $a \in \Sigma$, let $\delta'(R, a) = \big\{s \in S : \exists r \in R \big(s \in \delta(r, a)\big)\big\}$. Equivalently,

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a).$$

Also let $s_0' = \{s_0'\}$. We then see that $M = (\Sigma, S', \delta', s_0', F')$ recognises A.

Assume then that $\delta_0 \neq \varnothing$ for N. For each $Q \subseteq S$, let

$$E(Q) = \left\{s \in S \,:\, \exists n \in \mathbb{N}\, \exists r \in Q \left(s = \delta^n(r, \epsilon)\right)\right\}.$$

We then let

$$\delta'(Q, a) = \left\{s \in S \,:\, \exists r \in Q\, s \in E\big(\delta(r, a)\big)\right\}$$

and let $s_0' = E(\{s_0\})$. We similarly see that $M = (\Sigma, S', \delta', s_0', F')$ recognises A.

Therefore, the theorem holds. □

1.1.3 Regular Expressions and Regular Languages

Definition 3. Let Σ be an alphabet, and let $a \in \Sigma$. Then R is a **regular expression over** Σ if

(a) $R = \varnothing$,

(b) $R = \epsilon$,

(c) $R = a$,

(d) $R = R_1 \cup R_2$ wherein R_1 and R_2 are regular expressions over Σ,

(e) $R = R_1R_2$[2] wherein R_1 and R_2 are regular expressions over Σ, or

(f) $R = R_1^*$ wherein R_1 is a regular expression over Σ.

The language described by a regular expression is a **regular language**. Each regular expression describes a unique regular language, while a regular language may have multiple distinct regular expressions describing it. If R is a regular expression, we denote the regular language it describes by $L(R)$.

Let Σ be an alphabet, let $a \in \Sigma$, and let R, R_1, and R_2 be regular expressions over Σ. If $R = \varnothing$, then $L(R) = \varnothing$. If $R = \epsilon$, then $L(R) = \{\epsilon\}$. If $R = a$, then $L(R) = \{a\}$. If $R = R_1 \cup R_2$, then $L(R) = L(R_1) \cup L(R_2)$. If $R = R_1R_2$, then $L(R) = L(R_1)L(R_2)$[3]. If $R = R_1^*$, then $L(R) = L(R_1)^*$.

[2] R_1R_2 denotes the concatenation of R_1 and R_2.

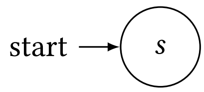
[3] If A and B are languages, AB denotes the concatenation of A and B, defined as $AB = \{ab \,:\, a \in A \wedge b \in B\}$.

1.1.4 Equivalence Between Finite-State Automata and Regular Languages

Lemma 1. *If a language is regular, then some nondeterministic finite-state automaton recognises it.*

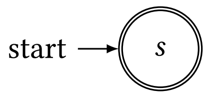
Proof. Let Σ be an alphabet, let A be a regular language over Σ, and let R be a regular expression which describes A.

If $R = \emptyset$, then the nondeterministic finite-state automaton N characterised by the following diagram recognises A.



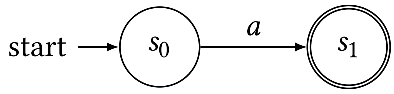
Equivalently, $N = (\Sigma, \{s\}, \delta, s, \emptyset)$ wherein $\delta(r, b) = \emptyset$ for any r and b.

If $R = \epsilon$, then the nondeterministic finite-state automaton N characterised by the following diagram recognises A.



Equivalently, $N = (\Sigma, \{s\}, \delta, s, \{s\})$ wherein $\delta(r, b) = \emptyset$ for any r and b.

If $R = a$ for some $a \in \Sigma$, then the nondeterministic finite-state automaton N characterised by the following diagram recognises it.



Equivalently, $N = (\Sigma, \{s_0, s_1\}, \delta, s_0, \{s_1\})$ wherein $\delta(s_0, a) = \{s_1\}$ and $\delta(r, b) = \emptyset$ if $r \neq s_0$ or $b \neq a$.

Assume that R_1 and R_2 are regular expressions over Σ, that $N_1 = (\Sigma, S_1, \delta_1, s_1, F_1)$ is a nondeterministic finite-state automaton recognising $L(R_1)$, and that $N_2 = (\Sigma, S_2, \delta_2, s_2, F_2)$ is a nondeterministic finite-state automaton recognising $L(R_2)$.

If $R = R_1 \cup R_2$, let s_0 be a state not in S_1 or S_2, let $S = S_1 \cup s_2 \cup \{s_0\}$, and let $F = F_1 \cup F_2$.

Define $\delta : S \times (\Sigma \cup \{\epsilon\}) \to \mathscr{P}(S)$ so that for each $r \in S$ and each $b \in \Sigma \cup \{\epsilon\}$ we have

$$\delta(r,b) = \begin{cases} \delta_1(r,b) & \text{if } r \in S_1, \\ \delta_2(r,b) & \text{if } r \in S_2, \\ \{s_1, s_2\} & \text{if } r = s_0 \land b = \epsilon, \text{ and} \\ \varnothing & \text{otherwise.} \end{cases}$$

We see that $N = (\Sigma, S, \delta, s_0, F)$ is a nondeterministic finite-state automaton recognising A.

If $R = R_1R_2$, let $S = S_1 \cup S_2$. Define $\delta : S \times (\Sigma \cup \{\epsilon\}) \to \mathscr{P}(S)$ so that for each $r \in S$ and each $b \in \Sigma \cup \{\epsilon\}$ we have

$$\delta(r,b) = \begin{cases} \delta_1(r,b) & \text{if } (r \in S_1 \land r \notin F_1) \lor (r \in F_1 \land b \neq \epsilon), \\ \delta_1(r,b) \cup \{s_2\} & \text{if } r \in F_1 \land b = \epsilon, \text{ and} \\ \delta_2(r,b) & \text{otherwise.} \end{cases}$$

We see that $N = (\Sigma, S, \delta, s_1, F_2)$ is a nondeterministic finite-state automaton recognising A.

If $R = R_1^*$, let s_0 be a state not in S_1, let $S = S_1 \cup \{s_0\}$, and let $F = F_1 \cup \{s_0\}$. Define $\delta : S \times (\Sigma \cup \{\epsilon\}) \to \mathscr{P}(S)$ so that for each $r \in S$ and each $b \in \Sigma \cup \{\epsilon\}$ we have

$$\delta(r,b) = \begin{cases} \delta_1(r,b) & \text{if } (r \in S_1 \land r \notin F_1) \lor (r \in F_1 \land b \neq \epsilon), \\ \delta_1(r,b) \cup \{s_1\} & \text{if } r \in F_1 \land b = \epsilon, \\ \{s_1\} & \text{if } r = s_0 \land b = \epsilon, \text{ and} \\ \varnothing & \text{otherwise.} \end{cases}$$

We see that $N = (\Sigma, S, \delta, s_0, F)$ is a nondeterministic finite-state automaton recognising A.

Therefore, the lemma holds by the principle of induction. □

Definition 4. A **generalised nondeterministic finite-state automaton** is an ordered quintuple $(\Sigma, S, \delta, s_0, s_1)$ wherein

(a) Σ is an alphabet,

(b) S is a finite set of states,

(c) $\delta : (S\setminus\{s_1\})\times(S\setminus\{s_0\}) \to \mathscr{R}$ wherein $\mathscr{R}$ is the set of all regular expressions over Σ is the transition function,

(d) $s_0 \in S$ is the initial state, and

(e) $s_1 \neq s_0 \in S$ is the accepting state.

Let $M = (\Sigma, S, \delta, s_0, s_1)$ be a generalised nondeterministic finite-state automaton and let w be a word over Σ. Then M accepts w if $w = w_0 \cdots w_n$ wherein $n \in \mathbb{N}$ such that each $w_i \in \Sigma^*$ for $i \in \mathbb{N}_{<n+1}$ and that there exists a sequence of states $(r_0, \ldots, r_{n+1})$ in S such that

(a) $r_0 = s_0$,

(b) $r_{n+1} = s_1$, and

(c) $w_i \in L(R_i)$ wherein $R_i = \delta(r_i, r_{i+1})$ for $i \in \mathbb{N}_{<n+1}$.

Lemma 2. *If a nondeterministic finite-state automaton recognises a language, then it is regular.*

Proof. Let $N = (\Sigma, S, \delta, s_0, F)$ be a nondeterministic finite-state automaton recognising the language A over Σ. We argue that A is described by some regular expression R over Σ.

Let $G = (\Sigma, S', \delta', s_0', s_1')$ be a generalised nondeterministic finite-state automaton such that

(a) $s_0' \notin S$,

(b) $s_1' \notin S$,

(c) $S' = S \cup \{s_0', s_1'\}$, and

(d) for each $r_0 \in S \cup \{s_0'\}$ and each $r_1 \in S \cup \{s_1'\}$ we have

$$\delta'(r_0, r_1) = \begin{cases} \epsilon & \text{if } (r_0 = s_0' \wedge r_1 = s_0) \vee (r_0 \in F \wedge r_1 = s_1'), \\ R' & \text{if } r_0 \in S \wedge r_1 \in S \wedge \forall r \in L(R') \big(r_1 \in \delta(r_0, r)\big), \text{ and} \\ \emptyset & \text{otherwise.} \end{cases}$$

We see that G also recognises A. We shall then convert G into regular expression R.

Let $k = |S'|$.

If $k = 2$, then $S' = \{s_0', s_1'\}$, and so $R = \delta'(s_0', s_1')$ is the regular expression.

If $k > 2$, let $s \in S'$ be distinct from s'_0 and s'_1, and let $G' = (\Sigma, S'', \delta'', s'_0, s'_1)$ be a generalised nondeterministic finite-state automaton such that

(a) $S'' = S' \setminus \{s\}$,

(b) for each $r_0 \in S'' \setminus \{s'_0\}$ and each $r_1 \in S'' \setminus \{s'_1\}$ we have

$$\delta''(r_0, r_1) = R_0 R_1^* R_2 \cup R_3$$

wherein $R_0 = \delta'(r_0, s)$, $R_1 = \delta'(s, s)$, $R_2 = \delta'(s, r_1)$, and $R_3 = \delta'(r_0, r_1)$.

We see that G' is equivalent to G.

Because G' has one fewer state than G, by the principle of induction, there exists regular expression R converted from G for any generalised nondeterministic finite-state automaton.

Therefore, the lemma holds. □

Theorem 2. *A language is regular if and only if some nondeterministic finite-state automaton recognises it.*

Proof. The theorem holds by Lemma 1 and Lemma 2. □

Corollary 1. *A language is regular if and only if some deterministic finite-state automaton recognises it.*

Proof. The corollary holds by Theorem 1 and Theorem 2. □

1.1.5 Nonregular Languages

Theorem 3 (pumping lemma). *If A is a regular language over Σ, then there is a $p \in \mathbb{Z}_{>0}$, the **pumping length**, such that if $w \in A$ is of length at least p, then there exist x, y, and $z \in \Sigma^*$ which satisfy*

(a) $w = xyz$,

(b) $xy^iz \in A$ for each $i \in \mathbb{N}$,

(c) $|y| > 0$, and

(d) $|xy| \leq p$.

Proof. Let $M = (\Sigma, S, \delta, s_0, F)$ be a deterministic finite-state automaton recognising A and let $p = |S|$.

Let $w = w_0 \cdots w_n$ wherein $n \in \mathbb{N}$ be a word in R of length $n + 1$ which satisfies $n + 1 \geq p$. Let $r_0, \dots, r_{n+1}$ be the sequence of states that M enters when accepting w. This sequence has length $n + 2$, which must be at least $p + 1$. Among the first $p + 1$ elements in the sequence, two must be the same state by the pigeonhole principle. Let the first of these be r_i and the second r_j. We note that $i \leq j - 1$ and that $j \leq p$. Now let $x = w_0 \cdots w_{i-1}$, $y = w_i \cdots w_{j-1}$, and $z = w_j \cdots w_n$.

Thus induced, $w = xyz$ satisfies the pumping lemma. □

Exercise 1. Let $\Sigma = \{0, 1\}$ be an alphabet. Prove that the language $A = \{0^n 1^n : n \in \mathbb{N}\}$ is not regular.

Solution. Assume for the sake of contradiction that A is regular. Let p be the pumping length thereof, and let $w = 0^p 1^p$. Then there exist words x, y, and $z \in A$ such that $w = xyz$, $xy^i z \in A$ for $i \in \mathbb{N}$, $|y| > 0$, and $|xy| \leq p$ by the pumping lemma. We argue that it is impossible that there exist such words.

We first see that $y = 0^j$ wherein $j \in \mathbb{Z}_{>0}$, for $|y| > 0$ and $|xy| \leq p$. Thus, $xyyz = 0^{p+j} 1^p \notin A$, which is a contradiction of $xy^i z \in A$ for $i \in \mathbb{N}$.

By the contradiction obtained above, the original proposition holds. ◊

1.2 Pushdown Automata and Context-Free Languages

1.2.1 Pushdown Automata

Definition 5. A **pushdown automaton** is an ordered sextuple $(\Sigma, \Gamma, S, \delta, s_0, F)$ wherein

(a) Σ is an alphabet, for the input,

(b) Γ is another alphabet, for the **stack**,

(c) S is a finite set of states,

(d) $\delta : S \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \to \mathscr{P}\big(S \times (\Gamma \cup \{\epsilon\})\big)$ is the transition function,

(e) $s_0 \in S$ is the initial state, and

(f) $F \subseteq S$ is the set of final or accepting states.

Let $M = (\Sigma, \Gamma, S, \delta, s_0, F)$ be a pushdown automaton and let w be a word over Σ. Then M accepts $w = w_0 \cdots w_n$ wherein $n \in \mathbb{N}$ such that $w_i \in \Sigma \cup \{\epsilon\}$ for $i \in \mathbb{N}_{<n+1}$ and that there exist a sequence of states $(r_0, \ldots, r_{n+1})$ in S and a sequence of words $(q_0, \ldots, q_{n+1})$ in Γ^* such that

(a) $r_0 = s_0$,

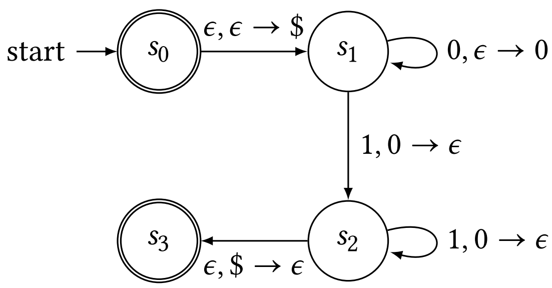
(b) $q_0 = \epsilon$,

(c) $(r_{i+1}, b) \in \delta(r_i, w_i, a)$, $q_i = at$, and $q_{i+1} = bt$ for some a and $b \in \Gamma \cup \{\epsilon\}$ and some $t \in \Gamma^*$ for $i \in \mathbb{N}_{<n+1}$, and

(d) $r_{n+1} \in F$.

EXERCISE 2. Let $\Sigma = \{0, 1\}$ be an alphabet. Construct a pushdown automaton which recognises the language $A = \{0^n 1^n : n \in \mathbb{N}\}$.

Solution. The pushdown automaton M characterised by the following diagram recognises A.



Equivalently, $M = (\Sigma, \Gamma, S, \delta, s_0, F)$ wherein

(a) $\Gamma = \{0, \$\}$,

(b) $S = \{s_0, s_1, s_2, s_3\}$,

(c) $F = \{s_0, s_3\}$, and

(d) for each $s \in S$, each $a \in \Sigma \cup \{\epsilon\}$, and each $b \in \Gamma \cup \{\epsilon\}$ we have

$$\delta(s, a, b) = \begin{cases} \{(s_1, \$)\} & \text{if } s = s_0 \wedge a = \epsilon \wedge b = \epsilon, \\ \{(s_1, 0)\} & \text{if } s = s_1 \wedge a = 0 \wedge b = \epsilon, \\ \{(s_2, \epsilon)\} & \text{if } (s = s_1 \vee s = s_2) \wedge a = 1 \wedge b = 0, \\ \{(s_3, \epsilon)\} & \text{if } s = s_2 \wedge a = \epsilon \wedge b = \$, \text{ and} \\ \emptyset & \text{otherwise} \end{cases}$$

is a pushdown automaton which recognises A. ◊

1.2.2 Context-Free Grammars and Context-Free Languagse

DEFINITION 6. A **context-free grammar** is an ordered quadruple (Σ, V, R, S) wherein

(a) Σ is an alphabet, the elements whereof are **terminals**,

(b) V is another alphabet, the elements whereof are **variables**, which is disjoint from Σ,

(c) $R : V \to (\Sigma \cup V)^*$ is a finite set of **production rules**, and

(d) $S \in V$ is the **start variable**.

Let (Σ, V, R, S) be a context-free grammar. If (A, w) wherein $A \in V$ and $w \in (\Sigma \cup V)^*$ is a production rule, we write $A \to w$. If u, v, and $w \in (\Sigma \cup V)^*$, and $A \to w$ is a production rule of the grammar, we say that uAv **yields** uwv, written $uAv \Rightarrow uwv$. We say that u **derives** v, written $u \Rightarrow^* v$, if $u = v$, $u \Rightarrow v$, or there exists a sequence $(u_0, \ldots, u_n)$ wherein $n \in \mathbb{N}$ such that

$$u \Rightarrow u_0 \Rightarrow \cdots \Rightarrow u_n \Rightarrow v.$$

If $A \to u$ and $A \to v$ are production rules of the grammar, we may denote them by $A \to u \mid v$. The **language of the grammar** is $\{w \in \Sigma^* : S \Rightarrow^* w\}$.

The language of a context-free grammar is a **context-free language**.

EXERCISE 3. Let $\Sigma = \{0, 1\}$ be an alphabet. Construct a context-free grammar which generates the language $A = \{0^n 1^n : n \in \mathbb{N}\}$.

Solution. Let (Σ, V, R, S) be the context-free grammar wherein $V = \{S\}$ and R consists of the following production rule

$$S \to 0S1 \mid \epsilon.$$

The language generated by the above context-free grammar is A. ◇

A derivation of a word in a context-free grammar is a **leftmost derivation** if at every step of production the leftmost remaining variable is the one substituted according to a production rule.

DEFINITION 7. A word is derived **ambiguously** in a context-free grammar if there exist two or more distinct leftmost derivations for it.

A context-free grammar is **ambiguous** is it generates some words ambiguously.

Some context-free languages can only be generated by ambiguous context-free grammars. Such languages are **inherently ambiguous**.

1.2.3 CHOMSKY NORMAL FORM

DEFINITION 8. A context-free grammar is **in Chomsky normal form** if every production rule thereof is

(a) $S \to \epsilon$ wherein S is the start variable,

(b) $A \to BC$ wherein A, B, and C are variables and B and C are not the start variable, or

(c) $A \to a$ wherein A is a variable and a is a terminal.

Theorem 4. *Any context-free language is generated by a context-free grammar in Chomsky normal form.*

Proof. Let (Σ, V, R, S) be a context-free grammar. We demonstrate a procedure to convert it into another context-free grammar in Chomsky normal form (Σ, V', R', S').

We first add $S' \to S$ as a production rule.

Second, if there exist rules of the form $A \to \epsilon$ wherein $A \neq S'$, we remove them and repeatedly replace any rule of the form $B \to uAv$ wherein $B \in V'$ and u and $v \in (\Sigma \cup V')^*$ with $B \to uv$ for each occurrence of A.

Third, if there exist rules of the form $A \to B$ wherein A and $B \in V'$, we remove them and replace any rule of the form $B \to u$ wherein $u \in (\Sigma \cup V')^*$ with $A \to u$.

Lastly, we replace each rule of the form $A \to u_0 \cdots u_n$ wherein $n \in \mathbb{N}$ and $u_i \in \Sigma \cup V'$ for $i \in \mathbb{N}_{<n+1}$ such that $n > 1$ with the rules $A \to u_0 A_0$, $A_0 \to u_1 A_1$, ..., $A_{n-2} \to u_{n-1} u_n$ and add A_i for $i \in \mathbb{N}_{<n-1}$ as variables. We then replace any terminal u_i for $i \in \mathbb{N}_{<n+1}$ with the new variable U_i while adding the rule $U_i \to u_i$.

The resultant context-free grammar is in Chomsky normal form, and therefore the theorem holds. □

1.2.4 Equivalence Between Pushdown Automata and Context-Free Languages

Lemma 3. *If a language is context-free, then some pushdown automaton recognises it.*

Proof. TODO □

Lemma 4. *If a pushdown automaton recognises a language, then it is context-free.*

Proof. TODO □

Theorem 5. *A language is context-free if and only if some pushdown automaton recognises it.*

Proof. The theorem holds by Lemma 3 and Lemma 4. □

Corollary 2. *Every regular language is context-free.*

Proof. Let Σ be an alphabet and let A be a regular language over Σ. Let $(\Sigma, S, \delta, s_0, F)$ be a nondeterministic finite-state automaton recognising A. Then the pushdown automaton $(\Sigma, \emptyset, S, \delta', s_0, F)$ wherein $\delta'(s, a, \epsilon) = \delta(s, a)$ for each $s \in S$ and each $a \in \Sigma \cup \{\epsilon\}$ also recognises A. Thus, A is context-free. □

1.2.5 Non-Context-Free Languages

Theorem 6 (pumping lemma for context-free languages). *If A is a context-free language over Σ, then there is a $p \in \mathbb{Z}_{>0}$, the pumping length, such that if $w \in A$ is of length at least p, then there exist u, v, x, y, and $z \in \Sigma^*$ which satisfy*

(a) $w = uvxyz$,

(b) $uv^ixy^iz \in A$ *for each* $i \in \mathbb{N}$,

(c) $|vy| > 0$, *and*

(d) $|vxy| \leq p$.

Proof. TODO □

1.2.6 Deterministic Pushdown Automata and Deterministic Context-Free Languages