

NOTES ON THE THEORY OF COMPUTATION

YANNAN MAO

7TH MARCH 2024

CONTENTS

1 Automata and Formal Languages	1
1.1 Finite-State Automata and Regular Languages	1
1.1.1 Deterministic Finite-State Automata	1
1.1.2 Nondeterministic Finite-State Automata	2
1.1.3 Regular Expressions and Regular Languages	3
1.1.4 Equivalence Between Finite-State Automata and Regular Languages	4
1.1.5 Nonregular Languages	7
1.2 Pushdown Automata and Context-Free Languages	8
1.2.1 Pushdown Automata	8
1.2.2 Context-Free Grammars and Context-Free Language	10
1.2.3 Chomsky Normal Form	11
1.2.4 Equivalence Between Pushdown Automata and Context-Free Languages	12
1.2.5 Non-Context-Free Languages	15
1.2.6 Deterministic Pushdown Automata and Deterministic Context-Free Languages	15

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 w as the concatenation $w_0 \cdots w_n$. If $w = a \cdots a$ wherein a is repeated n times for some $n \in \mathbb{Z}_{>0}$, we may write w as a^n . The empty word is denoted by ϵ , and for any element a of an alphabet a^0 is the empty word. The set of all words over Σ is Σ^* ¹. A *formal language over the alphabet* Σ is a subset of Σ^* . The attributive “formal” connotes that such languages lack semantics.

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 . Two automata are equivalent if they recognise the same language.

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 $(\Sigma, Q, \delta, q_0, F)$ wherein

- (a) Σ is an alphabet,
- (b) Q is a finite set of *states*,
- (c) $\delta : Q \times \Sigma \rightarrow Q$ is the *transition function*,
- (d) $q_0 \in Q$ is the *initial state*, and
- (e) $F \subseteq Q$ is the set of *accepting states*.

Let $M = (\Sigma, Q, \delta, q_0, F)$ 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 Q such that

¹* denotes the unary operator of Kleene star, defined as $A^* = \{a_0 \cdots a_n : n \in \mathbb{N} \wedge \forall i \in \mathbb{N}_{<n+1} (a_i \in A)\} \cup \{\epsilon\}$ for a subset A of an alphabet, and $a^* = \{a^n : n \in \mathbb{N}\}$ for an element a of an alphabet.

- (a) $r_0 = q_0$,
- (b) $\delta(r_i, w_i) = r_{i+1}$ for $i \in \mathbb{N}_{<n+1}$, and
- (c) $r_{n+1} \in F$.

Furthermore, M accepts ϵ if $q_0 \in F$.

1.1.2 NONDETERMINISTIC FINITE-STATE AUTOMATA

DEFINITION 2. A *nondeterministic finite-state automaton* is an ordered quintuple $(\Sigma, Q, \delta, q_0, F)$ wherein

- (a) Σ is an alphabet,
- (b) Q is a finite set of states,
- (c) $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$ is the transition function,
- (d) $q_0 \in Q$ is the initial state, and
- (e) $F \subseteq Q$ is the set of accepting states.

Let $N = (\Sigma, Q, \delta, q_0, F)$ be a nondeterministic finite-state automaton and let w be a word over Σ . Then N accepts w if $w = w_0 \cdots w_n$ wherein $n \in \mathbb{N}$ such that each $w_i \in \Sigma \cup \{\epsilon\}$ for some $i \in \mathbb{N}_{<n+1}$ and that there exists a sequence of states (r_0, \dots, r_{n+1}) in Q such that

- (a) $r_0 = q_0$,
- (b) $r_{i+1} \in \delta(r_i, w_i)$ for $i \in \mathbb{N}_{<n+1}$, and
- (c) $r_{n+1} \in F$.

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

Proof. Let Σ be an alphabet, let A be a language over Σ , and let $N = (\Sigma, Q, \delta, q_0, F)$ be a nondeterministic finite-state automaton recognising A . We construct a deterministic finite-state automaton $M = (\Sigma, Q', \delta', q'_0, F')$ which also recognises A .

We first see that $Q' = \mathcal{P}(Q)$ and that $F' = \{R \in Q' : R \cap F \neq \emptyset\}$.

Let $\delta_0 : Q \times \{\epsilon\} \rightarrow \mathcal{P}(Q)$ be defined as $\delta_0(q, \epsilon) = \delta(q, \epsilon)$ for each $q \in Q$. Assume first that, thus induced, $\delta_0 = \emptyset$ for N . For each $R \in Q'$ and each $a \in \Sigma$, let $\delta'(R, a) =$

$\{q \in Q : \exists r \in R (q \in \delta(r, a))\}$. Equivalently,

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

Also let $q'_0 = \{q'_0\}$. We then see that $M = (\Sigma, Q', \delta', q'_0, F')$ recognises A .

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

$$E(R) = \{q \in Q : \exists n \in \mathbb{N} \exists r \in R (q = \delta^n(r, \epsilon))\}.$$

We then let

$$\delta'(R, a) = \{q \in Q : \exists r \in R s \in E(\delta(r, a))\}$$

and let $q'_0 = E(\{q_0\})$. We similarly see that $M = (\Sigma, Q', \delta', q'_0, F')$ recognises A .

Therefore, the theorem holds. □

1.1.3 REGULAR EXPRESSIONS AND REGULAR LANGUAGES

DEFINITION 3. Let Σ be an alphabet. Then R is a *regular expression* over Σ if

- (a) $R = \emptyset$,
- (b) $R = \epsilon$,
- (c) $R = a$ for some $a \in \Sigma$,
- (d) $R = R_1 \cup R_2$ wherein R_1 and R_2 are regular expressions over Σ ,
- (e) $R = R_1 R_2$ 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*, which is unique.

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 $R = \emptyset$, then $L(R) = \emptyset$. 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 $R = R_1 R_2$, then $L(R) = L(R_1) L(R_2)$ ³. If $R = R_1^*$,

² $R_1 R_2$ denotes the concatenation of R_1 and R_2 .

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

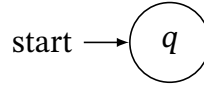
then $L(R) = L(R_1)^*$.

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.*

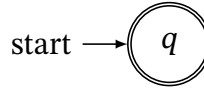
Proof. Let Σ be an alphabet and let R be a regular expression over Σ .

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



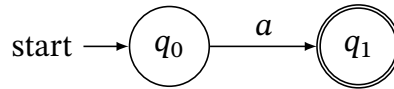
Equivalently, $N = (\Sigma, \{q\}, \delta, q, \emptyset)$ wherein $\delta(r, b) = \emptyset$ for any r and b .

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



Equivalently, $N = (\Sigma, \{q\}, \delta, q, \{q\})$ 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 $L(R)$.



Equivalently, $N = (\Sigma, \{q_0, q_1\}, \delta, q_0, \{q_1\})$ wherein $\delta(q_0, a) = \{q_1\}$ and $\delta(r, b) = \emptyset$ if $r \neq q_0$ or $b \neq a$.

Assume that R_1 and R_2 are regular expressions over Σ , that $N_1 = (\Sigma, Q_1, \delta_1, q_1, F_1)$ is a nondeterministic finite-state automaton recognising $L(R_1)$, and that $N_2 = (\Sigma, Q_2, \delta_2, q_2, F_2)$ is a nondeterministic finite-state automaton recognising $L(R_2)$.

If $R = R_1 \cup R_2$, let $\{q_0\}$ be disjoint from Q_1 and Q_2 , let $Q = Q_1 \cup Q_2 \cup \{q_0\}$, and let $F = F_1 \cup F_2$. Define $\delta: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$ so that for each $r \in Q$ and each $b \in \Sigma \cup \{\epsilon\}$

we have

$$\delta(r, b) = \begin{cases} \delta_1(r, b) & \text{if } r \in Q_1, \\ \delta_2(r, b) & \text{if } r \in Q_2, \\ \{q_1, q_2\} & \text{if } r = q_0 \wedge b = \epsilon, \text{ and} \\ \emptyset & \text{otherwise.} \end{cases}$$

We see that $N = (\Sigma, Q, \delta, q_0, F)$ is a nondeterministic finite-state automaton recognising $L(R)$.

If $R = R_1 R_2$, let $Q = Q_1 \cup Q_2$. Define $\delta: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$ so that for each $r \in Q$ and each $b \in \Sigma \cup \{\epsilon\}$ we have

$$\delta(r, b) = \begin{cases} \delta_1(r, b) & \text{if } (r \in Q_1 \wedge r \notin F_1) \vee (r \in F_1 \wedge b \neq \epsilon), \\ \delta_1(r, b) \cup \{q_2\} & \text{if } r \in F_1 \wedge b = \epsilon, \text{ and} \\ \delta_2(r, b) & \text{otherwise.} \end{cases}$$

We see that $N = (\Sigma, Q, \delta, q_1, F_2)$ is a nondeterministic finite-state automaton recognising $L(R)$.

If $R = R_1^*$, let $\{q_0\}$ be disjoint from Q_1 , let $Q = Q_1 \cup \{q_0\}$, and let $F = F_1 \cup \{q_0\}$. Define $\delta: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$ so that for each $r \in Q$ and each $b \in \Sigma \cup \{\epsilon\}$ we have

$$\delta(r, b) = \begin{cases} \delta_1(r, b) & \text{if } r \in Q_1 \setminus F_1 \vee (r \in F_1 \wedge b \neq \epsilon), \\ \delta_1(r, b) \cup \{q_1\} & \text{if } r \in F_1 \wedge b = \epsilon, \\ \{q_1\} & \text{if } r = q_0 \wedge b = \epsilon, \text{ and} \\ \emptyset & \text{otherwise.} \end{cases}$$

We see that $N = (\Sigma, Q, \delta, q_0, F)$ is a nondeterministic finite-state automaton recognising $L(R)$.

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

DEFINITION 4. A *generalised nondeterministic finite-state automaton* is an ordered quintuple $(\Sigma, Q, \delta, q_0, q_1)$ wherein

- (a) Σ is an alphabet,
- (b) Q is a finite set of states,
- (c) $\delta: (Q \setminus \{q_1\}) \times (Q \setminus \{q_0\}) \rightarrow \mathcal{R}$ wherein \mathcal{R} is the set of all regular expressions over Σ is the transition function,
- (d) $q_0 \in Q$ is the initial state, and
- (e) $q_1 \neq q_0 \in Q$ is the accepting state.

Let $G = (\Sigma, Q, \delta, q_0, q_1)$ 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 some $i \in \mathbb{N}_{<n+1}$ and that there exists a sequence of states (r_0, \dots, r_{n+1}) in Q such that

- (a) $r_0 = q_0$,
- (b) $r_{n+1} = q_1$, and
- (c) $w_i \in L(\delta(r_i, r_{i+1}))$ for $i \in \mathbb{N}_{<n+1}$.

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

Proof. Let Σ be an alphabet, let A be a language over Σ , and let $N = (\Sigma, Q, \delta, q_0, F)$ be a nondeterministic finite-state automaton recognising A . We argue that A is described by some regular expression R over Σ .

Let $G = (\Sigma, Q', \delta', q'_0, q'_1)$ be a generalised nondeterministic finite-state automaton such that

- (a) $\{q'_0, q'_1\} \cap Q = \emptyset$,
- (b) $Q' = Q \cup \{q'_0, q'_1\}$, and
- (c) for each $r_0 \in Q' \setminus \{q'_1\}$ and each $r_1 \in Q' \setminus \{q'_0\}$ we have

$$\delta'(r_0, r_1) = \begin{cases} \epsilon & \text{if } (r_0 = q'_0 \wedge r_1 = q_0) \vee (r_0 \in F \wedge r_1 = q'_1), \\ R' & \text{if } r_0 \in Q \wedge r_1 \in Q \wedge \forall r \in L(R') (r_1 \in \delta(r_0, r)), \text{ and} \\ \emptyset & \text{otherwise.} \end{cases}$$

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

Let $k = |Q'|$.

If $k = 2$, then $Q' = \{q'_0, q'_1\}$, and so $R = \delta'(q'_0, q'_1)$ is the regular expression.

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

- (a) $Q'' = Q' \setminus \{q\}$,
- (b) for each $r_0 \in Q'' \setminus \{q'_0\}$ and each $r_1 \in Q'' \setminus \{q'_1\}$ we have

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

wherein $R_0 = \delta'(r_0, q)$, $R_1 = \delta'(q, q)$, $R_2 = \delta'(q, r_1)$, 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). *Let Σ be an alphabet. If A is a regular language over Σ , then there is some $p \in \mathbb{Z}_{>0}$, the **pumping length**, such that if $w \in A$ satisfies $|w| \geq p$, 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 $M = (\Sigma, Q, \delta, q_0, F)$ be a deterministic finite-state automaton recognising A and let $p = |Q|$.

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. \square

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 x, y , and $z \in \Sigma^*$ such that $w = xyz$, that $xy^iz \in A$ for $i \in \mathbb{N}$, that $|y| > 0$, and that $|xy| \leq p$ 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 $xy^iz \in A$ for $i \in \mathbb{N}$.

By the contradiction obtained above, the original proposition holds. \diamond

1.2 PUSHDOWN AUTOMATA AND CONTEXT-FREE LANGUAGES

1.2.1 PUSHDOWN AUTOMATA

DEFINITION 5. A *pushdown automaton* is an ordered sextuple $(\Sigma, \Gamma, Q, \delta, q_0, F)$ wherein

- (a) Σ is an alphabet for the input,

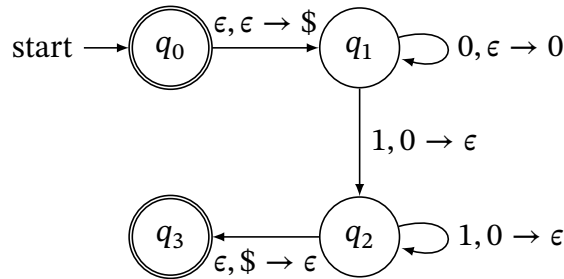
- (b) Γ is another alphabet for the *stack*,
- (c) Q is a finite set of states,
- (d) $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\epsilon\}))$ is the transition function,
- (e) $q_0 \in Q$ is the initial state, and
- (f) $F \subseteq Q$ is the set of accepting states.

Let $P = (\Sigma, \Gamma, Q, \delta, q_0, F)$ 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 some $i \in \mathbb{N}_{<n+1}$ and that there exist a sequence of states (r_0, \dots, r_{n+1}) in Q and a sequence of words (s_0, \dots, s_{n+1}) in Γ^* such that

- (a) $r_0 = q_0$,
- (b) $s_0 = \epsilon$,
- (c) for each $i \in \mathbb{N}_{<n+1}$ there exist some a and $b \in \Gamma \cup \{\epsilon\}$ and some $t \in \Gamma^*$ such that $(r_{i+1}, b) \in \delta(r_i, w_i, a)$, that $s_i = at$, and that $s_{i+1} = bt$, 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 P characterised by the following diagram recognises A .



Equivalently, $P = (\Sigma, \Gamma, Q, \delta, q_0, F)$ wherein

- (a) $\Gamma = \{0, \$\}$,
- (b) $Q = \{q_0, q_1, q_2, q_3\}$,
- (c) $F = \{q_0, q_3\}$, and

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

$$\delta(q, b, s) = \begin{cases} \{(q_1, \$)\} & \text{if } q = q_0 \wedge b = \epsilon \wedge s = \epsilon, \\ \{(q_1, 0)\} & \text{if } q = q_1 \wedge b = 0 \wedge s = \epsilon, \\ \{(q_2, \epsilon)\} & \text{if } (q = q_1 \vee q = q_2) \wedge b = 1 \wedge s = 0, \\ \{(q_3, \epsilon)\} & \text{if } q = q_2 \wedge b = \epsilon \wedge s = \$, \text{ and} \\ \emptyset & \text{otherwise} \end{cases}$$

is a pushdown automaton which recognises A . ◇

1.2.2 CONTEXT-FREE GRAMMARS AND CONTEXT-FREE LANGUAGE

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

- (a) Σ is an alphabet of *terminals*,
- (b) V is another alphabet of *variables*, which is disjoint from Σ ,
- (c) $R : V \rightarrow (\Sigma \cup V)^*$ 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 $R(A) = w$ wherein $A \in V$ and $w \in (\Sigma \cup V)^*$ is a production rule, we write $A \rightarrow w$. Let u, v , and $w \in (\Sigma \cup V)^*$. If $A \rightarrow w$ is a production rule, we say that uAv *yields* uwv and write $uAv \Rightarrow uwv$. We say that u *derives* v and write $u \Rightarrow^* v$ if $u = v$, $u \Rightarrow v$, or there exists a sequence (u_0, \dots, u_n) in $(\Sigma \cup V)^*$ for some $n \in \mathbb{N}$ such that

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

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

The language generated by 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

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

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* if 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) $S \rightarrow \epsilon$ wherein S is the start variable,
- (b) $A \rightarrow BC$ wherein A, B , and C are variables and B and C are not the start variable, or
- (c) $A \rightarrow a$ 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 $S' \rightarrow S$ as a production rule.

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

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

Lastly, we replace each rule of the form $A \rightarrow u_0 \cdots u_n$ 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 $A \rightarrow u_0A_0$, $A_0 \rightarrow u_1A_1$, ..., $A_{n-2} \rightarrow u_{n-1}u_n$ 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 $U_i \rightarrow u_i$.

The resultant context-free grammar is in Chomsky normal form, and thus the theorem holds. \square

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. Let Σ be an alphabet, let A be a context-free language over Σ , and let $G = (\Sigma, V, R, S)$ be a context-free grammar which generates A . We construct a pushdown automaton $P = (\Sigma, \Gamma, Q, \delta, q_0, F)$ which recognises A .

Let $b \in \Sigma \cup \{\epsilon\}$, let $s \in \Gamma \cup \{\epsilon\}$, and let q and $r \in Q$. Let $u = u_0 \cdots u_i$ wherein $i \in \mathbb{N}$ be a word over Γ . We denote by $(r, u) \in \delta(q, b, s)$ that there exist a sequence (q_0, \dots, q_{i-1}) in Q such that

- (a) $(q_0, u_i) \in \delta(q, b, s)$,
- (b) $\{(q_{j+1}, u_{i-j-1})\} = \delta(q_j, \epsilon, \epsilon)$ for $j \in \mathbb{N}_{<i-1}$, and
- (c) $\{(r, u_0)\} = \delta(q_{i-1}, \epsilon, \epsilon)$.

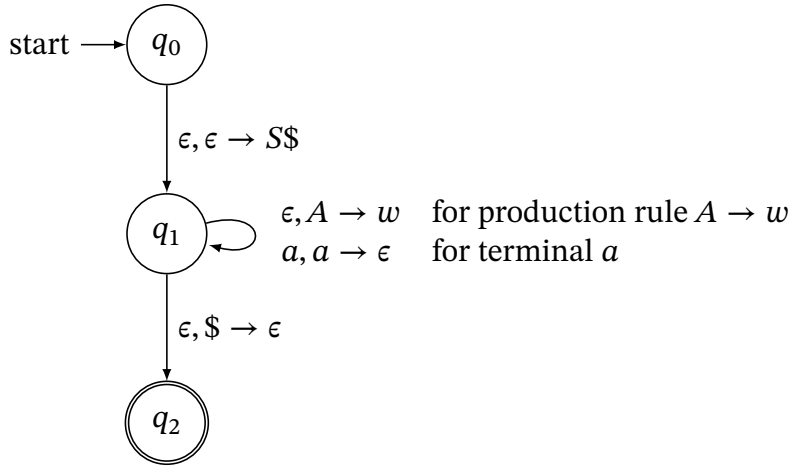
Let $Q = E \cup \{q_0, q_1, q_2\}$ and let $F = \{q_2\}$. Let $\{\$$ be disjoint from Σ and V , and let

$\Gamma = \Sigma \cup V \cup \{\$\}$. Let δ be defined as

$$\delta(q, b, s) = \begin{cases} \{(q_1, S\$)\} & \text{if } q = q_0 \wedge b = \epsilon \wedge s = \epsilon, \\ \{(q_1, w)\} & \text{if } q = q_1 \wedge b = \epsilon \wedge s = A \wedge (A \rightarrow w) \in R, \\ \{(q_1, \epsilon)\} & \text{if } q = q_1 \wedge b = a \wedge s = a \in \Sigma, \\ \{(q_2, \epsilon)\} & \text{if } q = q_1 \wedge b = \epsilon \wedge s = \$, \text{ and,} \\ \emptyset & \text{otherwise.} \end{cases}$$

Let $E \subseteq Q$ consist of those states necessary to make the δ as described above well-defined per the notation given in the previous paragraph.

The following diagram illustrates the constructed P .



Thus defined, the pushdown automaton P recognises A . Therefore, the lemma holds. □

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

Proof. Let $P = (\Sigma, \Gamma, Q, \delta, q_0, F)$ be a pushdown automaton. We construct a context-free grammar $G = (\Sigma, V, R, S)$ which generates all words over Σ accepted by P .

We first let $P' = (\Sigma, \Gamma, Q, \delta', q_0, F')$ be a pushdown automaton equivalent to P such that

- (a) $F' = \{q_1\}$,
- (b) there exist $q \in Q$, $b \in \Sigma \cup \{\epsilon\}$, and $s \in \Gamma \cup \{\epsilon\}$ which satisfy $\{q_1, \epsilon\} \in \delta'(q, b, s)$, and

- (c) if $\{r_1, s_1\} \in \delta(r_0, b, s_0)$ for some r_0 and $r_1 \in Q$, some $b \in \Sigma \cup \{\epsilon\}$, and some s_0 and $s_1 \in \Gamma \cup \{\epsilon\}$, then $s_0 = \epsilon$ or $s_1 = \epsilon$.

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 $(\Sigma, Q, \delta, q_0, F)$ be a nondeterministic finite-state automaton recognising A . Then the pushdown automaton $(\Sigma, \emptyset, Q, \delta', q_0, F)$ wherein $\delta'(q, b, \epsilon) = \delta(q, b)$ for each $q \in Q$ and each $b \in \Sigma \cup \{\epsilon\}$ also recognises A . Thus, A is context-free.

□

EXERCISE 4. Let $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ be an alphabet. Then $R = (1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)^*$ is a regular expression over Σ , and $L(R)$ is the set of positive integers in base 10 written in the Indo–Arabic numeral system.

Construct a context-free grammar which generates $L(R)$.

Solution. The context-free grammar (Σ, V, R, S) wherein $V = \{S, A, B\}$ and R consists of the production rules

$$S \rightarrow AB^*$$

$$A \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

$$B \rightarrow A \mid 0$$

generates $L(R)$.

◇

1.2.5 NON-CONTEXT-FREE LANGUAGES

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

- (a) $w = uvxyz$,
- (b) $uv^i xy^i z \in A$ 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