University of Houston

Notes

COSC 3340
Intro. to Automata and Computability

Ernst Leiss

Khalid Hourani

Contents

Chapter 1

Formal Languages

1.1 Regular Languages

1.1.1 Introduction

Definition 1.1.1.1. An **Alphabet** is a finite, non-empty set of atomic symbols.

Definition 1.1.1.2. A **word** or **string** is any finite sequence of symbols from an alphabet.

Definition 1.1.1.3. The **length** of a string, s, denoted $|s|$, is the number of symbols in s.

Definition 1.1.1.4. Given strings $s = s_1 s_2 \dots s_n$ and $t = t_1 t_2 \dots t_m$, their **concatenation** is defined

$$s \cdot t = s_1 s_2 \cdots s_n t_1 t_2 \cdots t_m$$

We denote by ε the **empty string**, the unique string of 0 characters.

Definition 1.1.1.5. Let A be any alphabet. The **Kleene Closure** of A, denoted A^*, is the set of all strings of any length over A.

Theorem 1.1.1.1. Let A be any finite set. Then A^* is countably infinite.

Proof. That A^* is infinite is straightforward: since A is non-empty, take $a \in A$. Then

$$\{a, aa, aaa, \dots\} \subseteq A^*$$

To see that it is countable, we first write $|A| = n$. Now, consider the set of all strings of length 0. This is simply $\{\varepsilon\}$. Moreover, there are n strings of length 1, n^2 strings of length 2, n^3 strings of length 3, and so on. Thus, we map ε to 0, the strings of length 1 to $1, 2, \dots, n$, the strings of length 2 to $n+1, n+2, \dots, n+n^2$, the strings of length 3 to $n+n^2+1, n+n^2+2, \dots, n+n^2+n^3$, and so on. This is a bijection from A^* to $\mathbb{N}$, which completes the proof. $\square$

Definition 1.1.1.6. Given an alphabet A, a **formal language** or simply **language** L is any subset of A^*.

Theorem 1.1.1.2. Given an alphabet A, the set of languages over A is uncountable.

Proof. Suppose, by way of contradiction, that the set of languages were countable, i.e., that we can enumerate the set as $\{L_1, L_2, L_3, \dots\}$. Consider the set of all strings $\{s_1, s_2, s_3, \dots\}$. Let L be the language defined as follows:

$$s_i \in L \text{ if and only if } s_i \notin L_i$$

To see that L is not in the above list, consider s_i. If s_i is in L, then s_i is not in L_i, by construction, and $L \neq L_i$. Similarly, if s_i is not in L, then s_i must be in L_i, by construction, and $L \neq L_i$. In other words, for all i, $L \neq L_i$. Then L is not in the above list, which is a contradiction. Hence, the set of languages is uncountable. $\square$

All set operations, such as union, intersection, complement, set-difference, etc. can be applied to languages, since languages are simply subsets of a Kleene Closure of an alphabet.

Definition 1.1.1.7. Given two languages L_1 and L_2, the concatenation $L_1 \cdot L_2$ is given by

$$L_1 \cdot L_2 = \{s \cdot t \mid s \in L_1 \text{ and } t \in L_2\}$$

Clearly, we have

$$\begin{aligned} L \cdot \emptyset &= \emptyset = \emptyset \cdot L \\ L \cdot \{\varepsilon\} &= L = \{\varepsilon\} \cdot L \end{aligned}$$

Note that $L_1 \cdot L_2$ is not the same as $L_1 \times L_2$. Let $L_1 = L_2 = \{\varepsilon, 0, 00\}$. Then

$$L_1 \times L_2 = \{(\varepsilon,\varepsilon), (\varepsilon,0), (\varepsilon,00), (0,\varepsilon), (0,0), (0,00), (00,\varepsilon), (00,0), (00,00)\}$$

whereas

$$L_1 \cdot L_2 = \{\varepsilon, 0, 00, 000, 0000\}$$

Definition 1.1.1.8. Given a language L, the **Kleene Closure** of L, L^*, is

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

where

$$L^i = \begin{cases} \{\varepsilon\} & \text{if } i = 0 \\ L \cdot L^{i-1} & \text{otherwise} \end{cases}$$

Note that, while 0^0 is normally left undefined, we define $\emptyset^0 = \{\varepsilon\}$.

Theorem 1.1.1.3. L^* is finite if and only if $L = \emptyset$ or $L = \{\varepsilon\}$.

Proof. If $L = \emptyset$, then $L^i = \emptyset^i = \emptyset$ for $i > 0$. Then

$$\begin{aligned} \emptyset^* &= \bigcup_{i=0}^{\infty} \emptyset^i \\ &= \emptyset^0 \cup \bigcup_{i=1}^{\infty} \emptyset^i \\ &= \{\varepsilon\} \cup \bigcup_{i=1}^{\infty} \emptyset \\ &= \{\varepsilon\} \end{aligned}$$

Similarly, if $L = \{\varepsilon\}$, then $L^i = \{\varepsilon\}$ for all i, and

$$\begin{aligned} \{\varepsilon\}^* &= \bigcup_{i=0}^{\infty} \{\varepsilon\}^i \\ &= \bigcup_{i=1}^{\infty} \{\varepsilon\} \\ &= \{\varepsilon\} \end{aligned}$$

However, if L is neither $\emptyset$ nor $\{\varepsilon\}$, then there exists a string $s \in L$ with length at least 1. Then $s, ss, sss, \ldots$, are in L^*, hence L^* is infinite. $\square$

1.1.2 Finite Automata

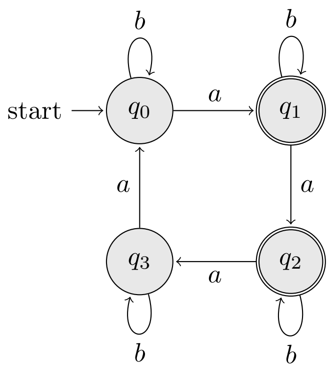
Definition 1.1.2.1. A **Deterministic Finite-State Automata** (DFA) or **Finite-State Machine** is a quintuple $(A, Q, \tau, q_0, \mathcal{F})$ where

A is the **alphabet**
Q is a finite, non-empty **set of states**
$\tau : Q \times A \to Q$ is the **transition function**
q_0 is the **initial state**
$\mathcal{F} \subseteq Q$ is the set of **final states**

We can extend τ as follows:
$\tau^* : Q \times A^* \to Q$

$$\tau^*(q, s) = \begin{cases} q & \text{if } s = \varepsilon \\ \tau^*(\tau(q, s_0), s') & \text{if } s = s_0 \cdot s' \end{cases}$$

We proceed informally and use τ to refer to τ^*.
Consider the following DFA:



The figure indicates that we begin at state q_0. The double-circles for states q_1 and q_2 indicate that they are accepting or final states. An arrow indicates the state to move to after receiving an input. For example, if we receive the input string *abba*, we begin at state q_0 and receive a, so we move to state q_1. We then receive b and stay in q_1. We repeat this for the next symbol, b, and then move to q_2 upon receiving the final a. Since q_2 is a final state, we say that this DFA **accepts** the string *abba*.

We can represent the above DFA using a table, as follows:

	a	b	
$\to q_0$	q_1	q_0	0
q_1	q_2	q_1	1
q_2	q_3	q_2	1
q_3	q_0	q_3	0

The first column indicates the states, while the first row indicates the symbols. The final column indicates whether a state is accepting: 0 refers to a non-final state, 1 to a final state. The remaining values indicate the transition function τ, e.g. $\tau(q_0, a) = q_1$, indicated by the entry corresponding to row q_0 and column a. Finally, the arrow pointing to q_0 indicates that it is the starting position.

Definition 1.1.2.2. Let $\underset{\sim}{D}$ be some DFA. Then $L(\underset{\sim}{D})$, the language accepted by the DFA, is

$$\{s \in A^* \mid \tau(q_0, s) \in \mathcal{F}\}$$

Definition 1.1.2.3. A language is **regular** if and only if there exists a DFA that accepts it.

Definition 1.1.2.4. A **Non-Deterministic Finite-State Automata** (NFA) is a quintuple

$$(A, Q, \tau, q_0, \mathcal{F})$$

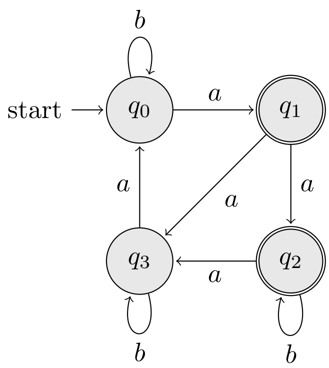
where

A is the **alphabet**
Q is a finite, non-empty **set of states**
$\tau : Q \times A \to 2^Q$ is the **transition function**
q_0 is the **initial state**
$\mathcal{F} \subseteq Q$ is the set of **final states**

We can extend τ as follows:
$\tau^* : 2^Q \times A^* \to 2^Q$

$$\tau^*(P, s) = \begin{cases} P & \text{if } s = \varepsilon \\ \tau^*\left(\bigcup_{q \in P} \tau(q, s_0), s'\right) & \text{if } s = s_0 \cdot s' \end{cases}$$

We proceed informally and use τ to refer to τ^*.
Consider the following NFA:



The diagrams for an NFA and DFA follow the same notation. However, the notation for the table differs slightly:

	a	b	
$\to q_0$	q_1	q_0	0
q_1	q_2q_3	$\emptyset$	1
q_2	q_3	q_2	1
q_3	q_0	q_3	0

The values of the transition function are now sets. We informally refer to the set $\{q_0\}$ by q_0, and similarly the set $\{q_2, q_3\}$ by q_2q_3. In some cases, to avoid ambiguity, we will use commas, e.g. we may represent $\{q_2, q_3\}$ as q_2, q_3. We similarly say, given a string s, if there exists a path through an NFA that ends in a final state, we say that the NFA **accepts** s.

Similarly, we define the set of languages accepted by an NFA $\underset{\sim}{N}$, $L(\underset{\sim}{N})$, as

$$L(\underset{\sim}{N}) = \{s \in A^* \mid \tau(q_0, s) \cap \mathcal{F} \neq \emptyset\}$$

It should be clear that each DFA is an NFA, but the reverse is not true. However, we can convert an NFA to a DFA on the powerset 2^Q by using the **subset construction**: begin with the initial state and traverse the NFA, adding unseen states to the left-most column until all paths have been exhausted. For example, with our NFA above, we begin with:

	a	b	
$\rightarrow q_0$	q_1	q_0	0

q_0 has already been seen, so we ignore it. q_1 is new, so we add it to the table:

	a	b
$\rightarrow q_0$	q_1	q_0
q_1		

We now visit the corresponding states of q_1, which are q_2q_3 and $\emptyset$, both of which have not yet been visited.

	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_2q_3	$\emptyset$
q_2q_3		
$\emptyset$		

When q_2 receives a, it transitions to state q_3. When q_3 receives a, it transitions to state q_0, so q_2q_3 transitions to q_0q_3. Similarly, q_2q_3 transitions to state q_2q_3 when it receives b.

	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_2q_3	$\emptyset$
q_2q_3	q_0q_3	q_2q_3
$\emptyset$		

The empty set transitions to the empty set, by definition.

	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_2q_3	$\emptyset$
q_2q_3	q_0q_3	q_2q_3
$\emptyset$	$\emptyset$	$\emptyset$

q_0q_3 has not yet been visited, so we add it to the left-most column:

	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_2q_3	$\emptyset$
q_2q_3	q_0q_3	q_2q_3
$\emptyset$	$\emptyset$	$\emptyset$
q_0q_3		

Then we visit its corresponding states:

	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_2q_3	$\emptyset$
q_2q_3	q_0q_3	q_2q_3
$\emptyset$	$\emptyset$	$\emptyset$
q_0q_3	q_0q_1	q_0q_3

Continuing, we end with the following DFA:

	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_2q_3	$\emptyset$
q_2q_3	q_0q_3	q_2q_3
$\emptyset$	$\emptyset$	$\emptyset$
q_0q_3	q_0q_1	q_0q_3
q_0q_1	$q_1q_2q_3$	q_0
$q_1q_2q_3$	$q_0q_2q_3$	q_2q_3
$q_0q_2q_3$	$q_0q_1q_3$	$q_0q_2q_3$
$q_0q_1q_3$	$q_0q_1q_2q_3$	q_0q_3
$q_0q_1q_2q_3$	$q_0q_1q_2q_3$	$q_0q_2q_3$

However, we need to include the accepting states. The accepting states of the NFA are q_1 and q_2, and thus any state including either state is accepting:

	a	b	
$\rightarrow q_0$	q_1	q_0	0
q_1	q_2q_3	$\emptyset$	1
q_2q_3	q_0q_3	q_2q_3	1
$\emptyset$	$\emptyset$	$\emptyset$	0
q_0q_3	q_0q_1	q_0q_3	0
q_0q_1	$q_1q_2q_3$	q_0	1
$q_1q_2q_3$	$q_0q_2q_3$	q_2q_3	1
$q_0q_2q_3$	$q_0q_1q_3$	$q_0q_2q_3$	1
$q_0q_1q_3$	$q_0q_1q_2q_3$	q_0q_3	1
$q_0q_1q_2q_3$	$q_0q_1q_2q_3$	$q_0q_2q_3$	1

Note that an NFA does not necessarily admit a DFA with as many states. Consider the following example:

	a	b	
$\rightarrow 0$	$\{1, 2, \ldots, n\}$	0	0
1	2	1	0
2	3	2	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
i	$i + 1$	i	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n - 1$	n	$n - 1$	0
n	1	n	1

The NFA above admits the following DFA:

	a	b	
$\rightarrow 0$	$\{1, 2, \ldots, n\}$	0	0
$\{1, 2, \ldots, n\}$	$\{1, 2, \ldots, n\}$	$\{1, 2, \ldots, n\}$	1

The above DFA contains only 2 states, despite the NFA containing $n + 1$ states.

That every NFA admits a DFA which accepts the same language shows that the class of languages denoted by DFAs, $\mathcal{L}_{\text{DFA}}$, is the same as the class of languages denoted by NFAs, $\mathcal{L}_{\text{NFA}}$, i.e, that

$$\mathcal{L}_{\text{DFA}} = \mathcal{L}_{\text{NFA}}$$

For an NFA, there is no guarantee of a unique smallest NFA which accepts the same strings. However, for a DFA, such a notion exists.

Consider two states, p and q, and corresponding L_p and L_q, where L_p has initial state p and L_q has initial state q. We say that p and q are distinguishable if there exists a string s such that s is in L_p and not in L_q, or vice-versa. We use this notion to **reduce** a DFA.

Begin with a partition of Q into subsets $\mathcal{F}$ and $Q - \mathcal{F}$, i.e., the accepting and rejecting states. For a pair of states p, q if the result of transitioning p and q falls into different partitions, we partition the subset and continue.

For example, given the following DFA:

	a	b	
$\to 0$	1	2	0
1	2	3	1
2	3	4	0
3	0	5	1
4	5	6	0
5	6	7	1
6	7	0	0
7	4	1	1

We have two partitions:

Rejecting	Accepting
0, 2, 4, 6	1, 3, 5, 7

Now, 0 gets sent to the accepting partition by a and to the rejecting partition by b. Similarly, 2, 4, and 6 get sent to the accepting partition by a and to the rejecting partition by b. Thus, they belong to the same partition.

In the same vein, 1 gets sent to the rejecting partition by a and to the accepting partition by b. Similarly, 3, 5, and 7 get sent to the rejecting partition by a and to the accepting partition by b. Thus, our next partition is

Rejecting	Accepting
0, 2, 4, 6	1, 3, 5, 7
0, 2, 4, 6	1, 3, 5, 7

That our row is the same as the preceding one indicates that we have finished, and now have a minimal DFA. Call the first subset p and the second q. When an element in p receives a, it is sent to q. When it receives b, it is sent to p. Similar logic for q gives our new DFA:

	a	b	
$\to p$	q	p	0
q	p	q	1

Recall that p began as a subset of the rejecting elements and q the accepting elements, which informs the last column of the above table.

Not all DFAs can be reduced. An obvious example is the above reduced DFA. For a less trivial example, consider the following DFA:

	a	b	
$\to 0$	1	2	0
1	2	3	1
2	3	4	0
3	0	5	1
4	5	6	0
5	6	7	1
6	7	0	0
7	4	2	1

Begin, as in the previous problem, with two partitions:

Rejecting	Accepting
0, 2, 4, 6	1, 3, 5, 7

As in the previous problem, 0, 2, 4, and 6 get sent to the same partition under a and b, respectively. Under a, 1, 3, 5, and 7 go to the rejecting partition. However, under b, 7 goes to the rejecting partition while 1, 3, and 5 go to the accepting partition, which means we must create a new partition for 7.

Rejecting	Accepting	
0, 2, 4, 6	1, 3, 5, 7	
0, 2, 4, 6	1, 3, 5	7

We continue the process, noting that there is no need to consider singletones, i.e., the partition $\{7\}$ is already in its finale state. Under a, 0, 2, and 4 get sent to the $\{1,3,5\}$ partition. Under b, they get sent to the $\{0,2,4,6\}$ partition. However, 6 gets sent to the $\{7\}$ partition, and so it must be partitioned separately. Similarly, 1 and 3 get sent to the $\{0,2,4,6\}$ partition under a, and to the $\{1,3,5\}$ partition under b. 5, on the other hand, gets sent to the $\{7\}$ partition, and must be partitioned separately. In total, we have:

Rejecting		Accepting		
0, 2, 4, 6		1, 3, 5, 7		
0, 2, 4, 6		1, 3, 5		7
0, 2, 4	6	1, 3	5	7

We continue:

Rejecting				Accepting			
0, 2, 4, 6				1, 3, 5, 7			
0, 2, 4, 6				1, 3, 5			7
0, 2, 4			6	1, 3		5	7
0, 2		4	6	1	3	5	7
0	2	4	6	1	3	5	7

Notice that the reduced DFA has 8 states, like the original! This means that the original DFA is already reduced, and cannot be reduced further.

1.1.3 Regular Expressions

Definition 1.1.3.1. Given an alphabet A, we define a **regular expression**

(a)
- $a \in A$ is a regular expression denoting the language $\{a\}$
- ε is a regular expression denoting $\{\varepsilon\}$
- $\emptyset$ is a regular expression denoting $\emptyset$

(b) If α and β are regular expressions denoting the languages $L(\alpha)$ and $L(\beta)$, respectively, then
- $\alpha \cup \beta$ denotes $L(\alpha) \cup L(\beta)$
- $\alpha \cdot \beta$ denotes $L(\alpha) \cdot L(\beta)$
- α^* denotes $L(\alpha)^*$

By convention, we define precedence of the operations $\cup$, $\cdot$, and * in that order. Thus,

$$b \cdot a^* \cup c = (b \cdot (a^*)) \cup c$$

A regular expression α over an alphabet A denotes the set of languages which accept α. Thus, we would like to construct an NFA $\underset{\sim}{N}$ such that $L(\underset{\sim}{N}) = L(\alpha)$.

The following NFA rejects all strings but a:

	a	$b \neq a$	
$\rightarrow q_0$	q_1	$\emptyset$	0
q_1	$\emptyset$	$\emptyset$	1

An NFA for only ε would appear as:

	$c \in A$	
$\rightarrow q_0$	$\emptyset$	1

And finally, an NFA for only $\emptyset$ is:

	$c \in A$	
$\rightarrow q_0$	$\emptyset$	0

Now, suppose we have an NFA for α and β. We wish to determine NFAs for $\alpha \cup \beta$, $\alpha \cdot \beta$, and α^*.

We define

$$\underset{\sim}{N}_\alpha = (A, Q_\alpha, \tau_\alpha, q_0, \mathcal{F}_\alpha)$$
$$\underset{\sim}{N}_\beta = (A, Q_\beta, \tau_\beta, q_0, \mathcal{F}_\beta)$$

such that

$$L(\underset{\sim}{N}_\alpha) = L(\alpha)$$
$$L(\underset{\sim}{N}_\beta) = L(\beta)$$
$$Q_\alpha \cap Q_\beta = \{q_0\}$$

and clarify that these automata are non-returning, i.e., that $q_0 \notin \tau(q_0, s)$ for any s of length 1 or greater.

We construct the **Union**

$$\underset{\sim}{N}_{\alpha\cup\beta} = (A, Q_{\alpha\cup\beta}, \tau_{\alpha\cup\beta}, q_0, \mathcal{F}_{\alpha\cup\beta})$$

where $Q_{\alpha\cup\beta} = Q_\alpha \cup Q_\beta$, $\mathcal{F}_{\alpha\cup\beta} = \mathcal{F}_\alpha \cup F_\beta$ and, for all $q \in Q_{\alpha\cup\beta}$ and $a \in A$

$$\tau_{\alpha\cup\beta}(q, a) = \begin{cases} \tau_\alpha(q_0, a) \cup \tau_\beta(q_0, a) & \text{if } q = q_0 \\ \tau_\alpha(q, a) & \text{if } q \in Q_\alpha - \{q_0\} \\ \tau_\beta(q, a) & \text{if } q \in Q_\beta - \{q_0\} \end{cases}$$

The **Concatenation** is constructed

$$\underset{\sim}{N}_{\alpha\beta} = (A, Q_{\alpha\beta}, \tau_{\alpha\beta}, q_0, \mathcal{F}_{\alpha\beta})$$

where $Q_{\alpha\beta} = Q_\alpha \cup Q_\beta$,

$$\mathcal{F}_{\alpha\beta} = \begin{cases} \mathcal{F}_\beta & \text{if } q_0 \notin \mathcal{F}_\beta \\ \mathcal{F}_\alpha \cup (\mathcal{F}_\beta - \{q_0\}) & \text{if } q_0 \in \mathcal{F}_\beta \end{cases}$$

and, for all $q \in Q_{\alpha\beta}$ and $a \in A$

$$\tau_{\alpha\beta}(q, a) = \begin{cases} \tau_\alpha(q, a) \cup \tau_\beta(q_0, a) & \text{if } q \in \mathcal{F}_\alpha \\ \tau_\alpha(q, a) & \text{if } q \in Q_\alpha - \mathcal{F}_\alpha \\ \tau_\beta(q, a) & \text{if } q \in Q_\beta - \{q_0\} \end{cases}$$

Finally, the **Kleene Closure** is constructed

$$\underset{\sim}{N}_{\alpha^*} = (A, Q_{\alpha^*}, \tau_{\alpha^*}, q_0, \mathcal{F}_{\alpha^*})$$

where $Q_{\alpha^*} = Q_\alpha$, $\mathcal{F}_{\alpha^*} = \mathcal{F}_\alpha \cup \{q_0\}$ and, for all $q \in Q_{\alpha^*}$ and $a \in A$

$$\tau_{\alpha^*}(q, a) = \begin{cases} \tau_\alpha(q, a) \cup \tau_\alpha(q_0, a) & \text{if } q \in \mathcal{F}_\alpha \\ \tau_\alpha(q, a) & \text{if } q \in Q_\alpha - \mathcal{F}_\alpha \end{cases}$$

This allows us to construct NFAs from a regular expression. Suppose we have a regular expression ab over $\{a, b\}$. Then we have

NFA for a

	a	b	
$\to q_0$	q_1	$\emptyset$	0
q_1	$\emptyset$	$\emptyset$	1

NFA for b

	a	b	
$\to q_0$	$\emptyset$	q_2	0
q_2	$\emptyset$	$\emptyset$	1

Applying the above construction for concatenation gives

NFA for ab

	a	b	
$\to q_0$	q_1	$\emptyset$	0
q_1	$\emptyset$	q_2	0
q_2	$\emptyset$	$\emptyset$	1

1.1.4 Solutions of Certain Language Equations

Given a regular expression, we can form an NFA which admits the same language by solving **Language Equations**. We show the following lemma before proceeding to examples:

Lemma 1. If $X = L \cdot X \cup M$ then $X = L^* \cdot M$ is a solution, and is unique if $\varepsilon \notin L$.

Proof. Clearly, $L^* \cdot M$ is a solution, since

$$L^* \cdot M = L \cdot (L^* \cdot M) \cup M$$

To prove uniqueness, suppose s_1 and s_2 are distinct solutions. There must exist a shortest-length string in s_1, say s. □

Consider the following NFA:

	a	b	
$\to 1$	2	1, 3	0
2	$\emptyset$	3	0
3	2, 3	1	1

This admits the following set of equations

$$X_1 = aX_2 \cup bX_1 \cup bX_3 \tag{1.1}$$
$$X_2 = bX_3 \tag{1.2}$$
$$X_3 = aX_2 \cup aX_3 \cup bX_1 \cup \varepsilon \tag{1.3}$$

We substitute (2) into (1) and (3):

$$X_1 = abX_3 \cup bX_1 \cup bX_3$$
$$X_3 = abX_3 \cup aX_3 \cup bX_1 \cup \varepsilon$$

which we rewrite as

$$X_1 = (ab \cup b)X_3 \cup bX_1$$
$$X_3 = (ab \cup a)X_3 \cup bX_1 \cup \varepsilon$$

We now apply our lemma to the equation for X_3

$$X_1 = (ab \cup b)X_3 \cup bX_1$$
$$X_3 = (ab \cup a)^*(bX_1 \cup \varepsilon)$$

We substitute X_3 into the equation for X_1

$$\begin{aligned}
X_1 &= (ab \cup b)(ab \cup a)^*(bX_1 \cup \varepsilon) \cup bX_1 \\
&= \big((ab \cup b)(ab \cup a)^* \cup b\big)\, X_1 \cup (ab \cup b)(ab \cup a)^* \cup bX_1 \\
&= \big((ab \cup b)(ab \cup a)^* \cup b\big)\, X_1 \cup (ab \cup b)(ab \cup a)^* \\
&= \big((ab \cup b)(ab \cup a)^* \cup b\big)^*(ab \cup b)(ab \cup a)^*
\end{aligned}$$

Consider the example:

	a	b	
$\rightarrow 1$	2	3	0
2	2	3	0
3	2	3	1

This admits the following system of equations:

$$\begin{aligned}
X_1 &= aX_2 \cup bX_3 \\
X_2 &= aX_2 \cup bX_3 \\
X_3 &= aX_2 \cup bX_3 \cup \varepsilon
\end{aligned}$$

From our lemma, we have $X_2 = a^*bX_3$:

$$\begin{aligned}
X_1 &= aa^*bX_3 \cup bX_3 \\
X_3 &= aa^*bX_3 \cup bX_3 \cup \varepsilon
\end{aligned}$$

which can be simplified:

$$\begin{aligned}
X_1 &= (aa^*b \cup b)X_3 \\
X_3 &= (aa^*b \cup b)X_3 \cup \varepsilon
\end{aligned}$$

Applying our lemma to X_3, we have

$$X_3 = (aa^*b \cup b)^*$$

Subsituting into X_1 gives

$$X_1 = (aa^*b \cup b)(aa^*b \cup b)^*$$

One final example:

	a	b	
$\rightarrow 1$	$\emptyset$	1, 2	1
2	1	$\emptyset$	1

$$\begin{aligned}
X_1 &= bX_1 \cup bX_2 \cup \varepsilon \\
X_2 &= aX_1 \cup \varepsilon
\end{aligned}$$

Substituting our equation for X_2 into X_1 gives

$$\begin{aligned}
X_1 &= bX_1 \cup b(aX_1 \cup \varepsilon) \cup \varepsilon \\
&= (b \cup ba)X_1 \cup b \cup \varepsilon \\
&= (b \cup ba)^*(b \cup \varepsilon)
\end{aligned}$$

1.1.5 Extended Regular Expressions

The languages we have discussed so far are **regular languages**. That is,

- Deterministic Finite Automaton
- Non-Deterministic Finite Automaton
- Regular Expression
- Solution of Languages Equations

are all regular languages. The following are **Closure Properties** of a regular language:

Theorem 1.1.5.1. Let $\mathcal{L}_\infty$ and $\mathcal{L}_\in$ be regular languages in some alphabet A. Then

1. $\mathcal{L}_1 \cup \mathcal{L}_2$
2. $\mathcal{L}_1 \cdot \mathcal{L}_2$
3. $\mathcal{L}_1^*$
4. $\overline{\mathcal{L}_1}$

are all regular languages in A.

Proof. 1, 2, and 3 follow from the definitions of regular expressions. For 4, consider a DFA $\underset{\sim}{D} = (A, Q, \tau, q_0, \mathcal{F})$ and consider any word $s \in A^*$. Further, let $\underset{\sim}{D}' = (A, Q, \tau, q_0, Q - \mathcal{F})$. If $w \in L(\underset{\sim}{D})$, then $w \notin L(\underset{\sim}{D}')$. On the other hand, if $w \notin L(\underset{\sim}{D})$, then $w \in L(\underset{\sim}{D}')$. Then $L(\underset{\sim}{D}') = \overline{L(\underset{\sim}{D})}$. □

This allows us to define the regular expression $\overline{\alpha}$:

Definition 1.1.5.1. Let α be any regular expression in some alphabet A. Then the regular expression $\overline{\alpha}$ is defined by

$$\overline{\alpha} = \overline{L(\alpha)}$$

If a regular expression contains a complement, it is an **extended regular expression**.

We can construct the DFA of the complement of a regular expression by finding the corresponding DFA and swapping the accepting and rejecting states. For example, consider the regular expression $\overline{01^*} \cap \overline{10^*}$ over $\{0, 1\}$.

$$\overline{01^*} \cap \overline{10^*} = \overline{\overline{01^*} \cup \overline{10^*}}$$

Similarly, we consider the example $\overline{(\overline{01^*}0)}^*$ over $\{0, 1, 2\}$.

It should be noted that the above process of swapping accepting and rejecting states *only works on a DFA*. Thus, if you wish to take the complement of an NFA, you must first convert it to a DFA.

1.1.6 Non-Regular Languages

Suppose we have a DFA

$$\underset{\sim}{D} = (A, Q, \tau, q_0, \mathcal{F})$$

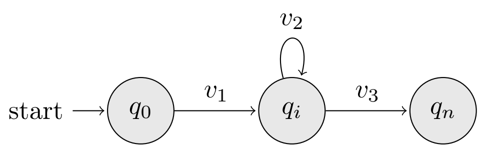
with $|Q| = n$. Consider the following set of equations

$$\begin{aligned}
q_1 &= \tau(q_0, a_1) \\
q_2 &= \tau(q_1, a_2) \\
&\vdots \\
q_i &= \tau(q_{i-1}, a_i) \\
&\vdots \\
q_{n-1} &= \tau(q_{n-2}, a_{n-1}) \\
q_n &= \tau(q_{n-1}, a_n)
\end{aligned}$$

Notice that we have $n+1$ states above, but only n states in Q. Then, by the Pidgeonhole Principle, there must be some state q_i that is visited twice. In other words, there exist an i, j with $i < j$ such that $q_i = q_j$. We then consider a string $s = a_1 a_2 \dots a_n$. Let $v_1 = a_1 a_2 \dots a_i$, $v_2 = a_{i+1} a_{i+2} \dots a_j$, and $v_3 = a_{j+1} a_{j+2} \dots a_n$. We see that

$$\tau(q_0, s) = \tau(q_0, v_1 v_2^k v_3)$$

for all $k \geq 0$.



Thus, we state **The Pumping Lemma** and provide a proof:

Theorem 1.1.6.1 (The Pumping Lemma)**.** Let L be any reguar language with corresponding DFA $(A, Q, \tau, q_0, \mathcal{F})$. Then there exists a $p > 0$ (called the **pumping length**) such that, for any string s of length p or longer, we can write $s = s_1 s_2 s_3$ and

- $|s_2| \geq 1$
- $|s_1 s_2| \leq p$
- $\tau(q_0, s) = \tau(q_0, s_1 s_2^n s_3)$ for all $n \geq 0$

We can use the above theorem to prove that certain languages are not regular.

Theorem 1.1.6.2. The language given by

$$L = \{a^i b^i \mid i \geq 0\}$$

is not regular.

Proof. Suppose, by way of contradiction, that L is regular with pumping length p. Consider the string $s = a^p b^p$. By the pumping lemma, we can write $s = s_1 s_2 s_3$ with $|s_1 s_2| \leq p$ and $|s_2| \geq 1$. Then $s_1 = a^p - k$ and $s_2 = a^k$ for some $1 \leq k \leq p$. Further, we have

$$\begin{aligned} s_1 s_2^n s_3 &= a^{p-k} (a^k)^n b^p \\ &= a^{p-k} a^{kn} b^p \\ &= a^{p+(n-1)k} b^p \in L \end{aligned}$$

Taking $n = 2$ gives $a^{p+k} b^p \in L$, a contradiction. Thus, L is not regular. $\square$

1.2 Context-Free Languages

1.2.1 Context-Free Grammars

Definition 1.2.1.1. A **Context-Free Grammar** is a quartuple $G = (N, T, P, S)$ where

N is a finite, non-empty set of variables (also called non-terminals)
T is an alphabet of terminals
$P \subseteq N \times (N \cup T)^*$ is a finite set of productions
$S \in N$ is the starting symbol

For any $(A, \gamma) \in P$, we write $A \to \gamma$, and say A **produces** γ.

By convention, we use upper-case letters to denote variables, lower-case to denote terminals and strings over the terminals, and Greek letters to denote strings over variables and terminals.

Definition 1.2.1.2. Given strings α and β, we say α **derives** β if there exist $A, \alpha_1, \alpha_2, \gamma$ such that

$$\begin{aligned}\alpha &= \alpha_1 A \alpha_2 \\ \beta &= \alpha_1 \gamma \alpha_2 \\ A &\to \gamma \in P\end{aligned}$$

and we write this $\alpha \Rightarrow \beta$.

We can define the language of a context-free grammar:

Definition 1.2.1.3. Given a context-free grammar G, the corresponding **context-free language** is

$$L(G) = \{w \mid S \Rightarrow w\}$$

Theorem 1.2.1.1. Every regular language is a context-free language.

Proof. Let L be a regular language and let $N = (Q, A, \tau, q_0, \mathcal{F})$ be its corresponding minimum DFA. □

However, not every context-free language is regular. For example, the language $\{a^n b^n \mid n \geq 0\}$ is not regular, but is a context-free language given by

1.2.2 Preprocessing a CFG

For any CFG G, we preprocess the language:

- Eliminate useless symbols
- Eliminate ε productions: $A \to \varepsilon$
- Eliminate unit productions: $A \to B$

Eliminating Useless Symbols

For example, suppose G is a context-free grammar given by:

$$\begin{aligned}S &\to aSb \mid cAd \\ A &\to aSc \mid bAd\end{aligned}$$

Then $L(G) = \emptyset$, since every terminal produces a string with a terminal. Thus, we can elminate S and A. In general, for any context-free grammar G, if no string s exists such that $A \Rightarrow s$, then we can eliminate A.

Consider the following example:

$$\begin{aligned}S &\to aS \mid bA \mid \varepsilon \\ A &\to cAA \mid dBB \\ B &\to aBA \mid bAA \mid cAC \\ C &\to aCb \mid S\end{aligned}$$

Notice that S produces ε, and so we cannot eliminate it. Similarly, C produces S, so we cannot eliminate it. At this point, it should be apparent that A and B do not produce terminals, and therefore can be eliminated. Further, we can eliminate terminals c and d since they are not involved in the productions of C or S. Graphically, we have

S
A
B
C

Now note that C cannot be reached from S, the starting state. Thus, we can eliminate C, and similarly eliminate b. Thus, our grammar can be reduced to

$$S \to aS \mid \varepsilon$$

To summarize: if a terminal string cannot be reached from a variable, or a variable cannot be reached from the starting symbol, it can be eliminated.

Eliminating ε Productions

To remove ε productions, we find the nullable non-terminals (variables). These are the non-terminals from which ε can be derived. Specifically, a variable A is nullable if it is of the form

$$A \to \varepsilon$$

or

$$A \to A_1 A_2 A_3 \dots A_n$$

where each A_i is nullable. Then, simply replace each *combination of nullable variables* with ε and eliminate ε from the right-hand side.

For example, consider the grammar

$$\begin{aligned}
S &\to ABCd \\
A &\to BC \\
B &\to bB \mid \varepsilon \\
C &\to cC \mid \varepsilon
\end{aligned}$$

Clearly, B and C are nullable. Then A is nullable because it produces BC. We then have

$$\begin{aligned}
S &\to ABCd \mid \cancel{A}BCd \mid A\cancel{B}Cd \mid AB\cancel{C}d \mid \cancel{A}\cancel{B}Cd \mid \cancel{A}B\cancel{C}d \mid A\cancel{B}\cancel{C}d \mid \cancel{A}\cancel{B}\cancel{C}d \\
A &\to BC \mid \cancel{B}C \mid B\cancel{C} \mid \cancel{B}\cancel{C} \\
B &\to bB \mid b\cancel{B} \\
C &\to cC \mid c\cancel{C}
\end{aligned}$$

which becomes

$$\begin{aligned}
S &\to ABCd \mid BCd \mid ACd \mid ABd \mid Cd \mid Bd \mid Ad \mid d \\
A &\to BC \mid C \mid B \\
B &\to bB \mid b \\
C &\to cC \mid c
\end{aligned}$$

Eliminating Unit Productions

To eliminate a unit production, simply replace any unit production

$$A \to B$$

with the productions for B. For example, consider the following grammar:

$$\begin{aligned}
S &\to Aa \mid B \\
A &\to b \mid B \\
B &\to A \mid a
\end{aligned}$$

We see that $S \to B$, $A \to B$, and $B \to A$ are unit rules, and replace them

$$\begin{aligned}
S &\to Aa \mid A \mid a \\
A &\to b \mid A \mid a \\
B &\to b \mid B \mid a
\end{aligned}$$

Similarly, consider the following example:

$$\begin{aligned} S &\to A \mid SS \\ A &\to B \mid AA \\ B &\to S \mid a \end{aligned}$$

1.2.3 Normal Forms

Chomsky Normal Form

Definition 1.2.3.1. A context-free language G is in **Chomsky Normal Form**[1] if all of its productions are of the form

$$\begin{aligned} A &\to BC \\ A &\to a \end{aligned}$$

Theorem 1.2.3.1. If G is a context-free grammar, there exists a Chomsky Normal Form grammar for $L(G) - \{\varepsilon\}$

Proof. □

To convert to Chomsky Normal Form, first preprocess the CFG. Then, for any production containing a non-solitary terminal, a, replace a with X_a, where $X_a \to a$. After this, we replace every production

$$A \to X_1 X_2 \dots X_n$$

with the following productions

$$\begin{aligned} A &\to X_1 A_1 \\ A_1 &\to X_2 A_2 \\ A_2 &\to X_3 A_3 \\ &\vdots \\ A_{n-2} &\to X_{n-1} X_n \end{aligned}$$

Greibach Normal Form

Definition 1.2.3.2. A context-free language is in **Greibach Normal Form**[2] if all of its productions are of the form

$$A \to a A_1 A_2 \dots A_n$$

To convert a CFG to Greibach Normal Form, you must remove left-recursion. First, preprocess the CFG. We can remove any *direct recursion* of the form

$$A \to A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \mid \dots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \beta_3 \mid \dots \mid \beta_n$$

where no β_i begins with A, by replacing the productions for A with

$$\begin{aligned} A &\to \beta_1 \mid \beta_2 \mid \beta_3 \mid \dots \mid \beta_n \mid \beta_1 A' \mid \beta_2 A' \mid \beta_3 A' \mid \dots \mid \beta_n A' \\ A' &\to \alpha_1 \mid \alpha_2 \mid \alpha_3 \mid \dots \mid \alpha_m \mid \alpha_1 A' \mid \alpha_2 A' \mid \alpha_3 A' \mid \dots \mid \alpha_m A' \end{aligned}$$

More generally, we eliminate *indirect recursion* by ordering the non-terminals and performing the following algorithm:

---

[1] This is also called **Chomsky Reduced Form**. Sometimes, Chomsky Normal Form includes the production $S \to \varepsilon$. In our usage, CNF and CRF are identical and do not include a ε production.

[2] Sometimes, the production $S \to \varepsilon$ is incuded. In our use, there is no ε production.

Algorithm Returns CFG with left-recursion removed

procedure S(G)
    for each non-terminal A_i **do**
        for each production $A_i \to \alpha_i$ **do**
            if α_i begins with A_j for $j < i$ **then**
                let β_i be α_i without the leading A_j
                remove the production $A_i \to \alpha_i$
                for each production $A_j \to \alpha_j$ **do**
                    Add rule $A_i \to \alpha_j\beta_i$
                end for
            end if
        end for
        remove direct left recursion for A_i
    end for
end procedure

Consider the following example:

$$\begin{aligned}
S &\to AS \mid a \\
A &\to BS \mid b \\
B &\to CS \mid c \\
C &\to SS \mid d
\end{aligned}$$

There is no preprocessing to do, since there are no useless or nullable variables and no unit productions. Notice that C includes indirect left-recursion. Thus, we write

$$\begin{aligned}
C &\to ASS \mid aS \mid d \\
&\to BSSS \mid bSS \mid aS \mid d \\
&\to CSSSS \mid cSSS \mid bSS \mid aS \mid d
\end{aligned}$$

Now, we eliminate the direct left-recursion from C

$$\begin{aligned}
C &\to cSSS \mid bSS \mid aS \mid d \mid cSSSC' \mid bSSC' \mid aSC' \mid dC' \\
C' &\to SSSS \mid SSSSC'
\end{aligned}$$

Next, we substitute the productions for C into B

$$B \to cSSSS \mid bSSS \mid aSS \mid dS \mid cSSSC'S \mid bSSC'S \mid aSC'S \mid dC'S \mid c$$

And similarly for A and S

$$\begin{aligned}
A &\to cSSSSS \mid bSSSS \mid aSSS \mid dSS \mid cSSSC'SS \mid bSSC'SS \mid aSC'SS \mid dC'SS \mid cS \mid b \\
S &\to cSSSSSS \mid bSSSSS \mid aSSSS \mid dSSS \mid cSSSC'SSS \mid bSSC'SSS \mid aSC'SSS \mid dC'SSS \mid cSS \mid bS \mid a
\end{aligned}$$

And finally for C'

$$\begin{aligned}
C' \to\ & cSSSSSSSSS \mid bSSSSSSSS \mid aSSSSSSS \mid dSSSSSS \mid cSSSC'SSSSSS \\
& \mid bSSC'SSSSSS \mid aSC'SSSSSS \mid dC'SSSSSS \mid cSSSSS \mid bSSSS \\
& \mid aSSSC' \mid cSSSSSSSSSC' \mid bSSSSSSSSC' \mid aSSSSSSSC' \mid dSSSSSSC' \mid cSSSC'SSSSSSC' \\
& \mid bSSC'SSSSSSC' \mid aSC'SSSSSSC' \mid dC'SSSSSSC' \mid cSSSSSC' \mid bSSSSC' \mid aSSSC'
\end{aligned}$$

Consider similarly the example:

$$\begin{aligned} S &\to AA \\ A &\to BaA \mid a \\ B &\to SBA \mid b \end{aligned}$$

Once again, there is no pre-processing to do. We see that B involves indirect left-recursion. Thus, we write

$$\begin{aligned} B &\to AABA \mid b \\ &\to BaAABA \mid aABA \mid b \end{aligned}$$

This involves direct left-recursion, so we write

$$\begin{aligned} B &\to aABA \mid b \mid aABAB' \mid bB' \\ B' &\to aAABA \mid aAABAB' \end{aligned}$$

We then substitute the productions for B into A

$$A \to aABAaA \mid baA \mid aABAB'aA \mid bB'aA \mid a$$

and finally substitute the productions for A into S

$$S \to aABAaAA \mid baAA \mid aABAB'aAA \mid bB'aAA \mid aA$$

In total, our GNF is

$$\begin{aligned} S &\to aABAaAA \mid baAA \mid aABAB'aAA \mid bB'aAA \mid aA \\ A &\to aABAaA \mid baA \mid aABAB'aA \mid bB'aA \mid a \\ B &\to aABA \mid b \mid aABAB' \mid bB' \\ B' &\to aAABA \mid aAABAB' \end{aligned}$$

1.2.4 Pumping Lemma for Context-Free Languages

Like with regular languages, there exists a pumping lemma for context-free languages.

Theorem 1.2.4.1. Let L be a context-free language. Then there exists a p such that for all strings s such that $|s| \geq p$, there exist strings s_1, s_2, s_3, s_4, s_5 such that

- $|s_2 s_4| \geq 1$
- $|s_2 s_3 s_4| \leq p$
- $s_1 s_2^n s_3 s_4^n s_5 \in L$ for all $n \geq 0$

Proof. □

We can use this to show that certain languages are not context-free. For example,

Theorem 1.2.4.2. $L = \{a^n b^n c^n \mid n \geq 1\}$ is not a context-free language.

Proof. Suppose, by way of contradiction, that L is a context-free language with pumping length p. Without loss of generality, suppose it is in Chomsky Normal Form. Consider the string $s = a^p b^p c^p$. Write $s = s_1 s_2 s_3 s_4 s_5$ with $|s_2 s_4| \geq 1$ and $|s_2 s_3 s_4| \leq p$. We consider $s_2 s_3 s_4$: clearly, it cannot contain a, b, *and* c, for then it would have length greater than p. Thus, $s_2 s_3 s_4$ must be one of the following

1. $s_2 s_3 s_4 = a^i$

2. $s_2s_3s_4 = a^ib^j$
3. $s_2s_3s_4 = b^i$
4. $s_2s_3s_4 = b^ic^j$
5. $s_2s_3s_4 = c^i$

We prove each case separately:

1. Write $s_1 = a^x$, $s_2 = a^y$, $s_3 = a^z$, $s_4 = a^w$. Then $s_5 = a^{p-(x+y+z+w)}b^pc^p$ At least one of y and w is non-zero. Then, by the pumping lemma,

$$\begin{aligned} s_1s_2^ns_3s_4^ns_5 &= a^xa^{ny}a^za^{nw}a^{p-(x+y+z+w)}b^pc^p \\ &= a^{p+(n-1)(y+w)}b^pc^p \end{aligned}$$

The above is clearly not in L for any $n \geq 2$.

2. There are two cases:

   (a) s_2 contains b

   (b) s_2 does not contain b

   In case (a), we have $s_1 = a^x$, $s_2 = a^{p-x}b^y$, $s_3 = b^z$, $s_4 = b^w$, and $s_5 = b^{p-(y+z+w)}c^p$. Pumping gives

$$\begin{aligned} s_1s_2^ns_3s_4^ns_5 &= a^xa^{n(p-x)}b^{ny}b^zb^{nw}b^{p-(y+z+w)}c^p \\ &= a^{n(p-x)+x}b^{p+(n-1)(y+w)}c^p \end{aligned}$$

   For $n \geq 2$, there are more bs than cs, so the above string is not in L. A contradiction.

   In case (b), we have $s_1 = a^x$, $s_2 = a^y$, $s_3 = a^zb^w$, $s_4 =$

$\square$

1.2.5 Closure Properties

Like regular languages, there are closure properties for context-free languages.

Theorem 1.2.5.1. If L_1 and L_2 are context-free languages with grammars $G_1 = (N_1, T, P_1, S_1)$ and $G_2 = (N_2, T, P_2, S_2)$, respectively, then

- $L_1 \cup L_2$
- L_1L_2
- L_1^*

are all context-free languages.

1.2.6 Pushdown Automata

Definition 1.2.6.1. A **Pushdown Automaton** is a septuple

$$\underset{\sim}{P} = (Q, T, \Gamma, \delta, q_0, Z_0, F)$$

where

Q is a finite non-empty set of states
T is an alphabet of input symbols
Γ is an alphabet of stack symbols
$\delta : Q \times (T \cup \{\varepsilon\}) \times \Gamma \to Q \times \Gamma^*$ is the move function
$q_0 \in Q$ is the initial state
$Z_0 \in \Gamma$ is the initial stack symbol
$F \subseteq Q$ is the set of accepting states

We define $(q, a, \gamma) \vdash (q', a', \gamma')$ when $\delta(q, a, \gamma) = (q', a', \gamma')$.

Informally, a pushdown automaton differs from a finite state machine in that it has a stack that allows it to:

1. Choose a transition based on the top of the stack
2. Push to or pop from the stack during a transition

Definition 1.2.6.2. Let $\underset{\sim}{P} = (Q, T, \Gamma, \delta, q_0, Z_0, F)$ be a pushdown automaton. The **language accepted by final state** is given by

$$L(\underset{\sim}{P}) = \{w \in T^* \mid (q_0, w, z_0) \vdash^* (f, \varepsilon, \gamma),\ \text{where } f \in F \text{ and } \gamma \in \Gamma^*\}$$

The **language accepted by empty stack** is given by

$$N(\underset{\sim}{P}) = \{w \in T^* \mid (q_0, w, z_0) \vdash^* (q, \varepsilon, \varepsilon),\ \text{where } q \in Q\}$$

where $\vdash^*$ denotes 0 or more transitions.

Consider the following example:

		a	b	ε
q_0	z_0	(q_0, z_0z)	$\emptyset$	$\emptyset$
	z	(q_0, zz)	(q_1, ε)	$\emptyset$
q_1	z_0	$\emptyset$	$\emptyset$	(q_f, ε)
	z	$\emptyset$	(q_1, ε)	$\emptyset$
q_f	z_0		accept	
	z			

This describes the language $L = \{a^n b^n \mid n \geq 1\}$
Consider the following language on $\{0, 1\}$:

$$L = \{w2w^R \mid w \in \{0, 1\}^*\}$$

where w^R is the reflection of w. This is given by the context-free grammar

$$S \to 0S0 \mid 1S1 \mid 2$$

This can also be defined by the following pushdown automaton:

		0	1	2	ε
q_0	z_0	(q_0, z_0z)	$\emptyset$	$\emptyset$	
	z	(q_0, zz)	(q_1, ε)	$\emptyset$	
q_1	z_0	$\emptyset$	$\emptyset$	(q_f, ε)	
	z	$\emptyset$	(q_1, ε)	$\emptyset$	
q_f	z_0		accept		
	z				

Theorem 1.2.6.1. For any pushdown automaton M, there exist pushdown automata P and Q such that $L(M) = N(P)$ and $N(L) = L(Q)$.

Chapter 2

Exercise Sets

2.1 Exercise Set 1

Exercise 1: Construct DFAs for the following NFAs using the subset construction:

(a)

	a	
$\rightarrow 1$	2	0
2	3	0
3	4	0
4	5	0
5	6	0
6	7	0
7	1, 2	1

(b)

	a	b	c	
$\rightarrow 1$	2	2	2	1
2	3	1	1, 2	1
3	4	3	$\emptyset$	1
4	5	4	4	1
5	1	5	5	1

(c)

	a	b	c	
$\rightarrow 1$	2	2	2	1
2	3	1	2, 3	1
3	4	3	$\emptyset$	1
4	5	4	4	1
5	1	5	5	1

Solution.

(a)

	a	
$\to$ 1	2	0
2	3	0
3	4	0
4	5	0
5	6	0
6	7	0
7	1, 2	1
1, 2	2, 3	0
2, 3	3, 4	0
3, 4	4, 5	0
4, 5	5, 6	0
5, 6	6, 7	0
6, 7	1, 2, 7	1
1, 2, 7	1, 2, 3	1
1, 2, 3	2, 3, 4	0
2, 3, 4	3, 4, 5	0
3, 4, 5	4, 5, 6	0
4, 5, 6	5, 6, 7	0
5, 6, 7	1, 2, 6, 7	1
1, 2, 6, 7	1, 2, 3, 7	1
1, 2, 3, 7	1, 2, 3, 4	1
1, 2, 3, 4	2, 3, 4, 5	0
2, 3, 4, 5	3, 4, 5, 6	0
3, 4, 5, 6	4, 5, 6, 7	0
4, 5, 6, 7	1, 2, 5, 6, 7	1
1, 2, 5, 6, 7	1, 2, 3, 6, 7	1
1, 2, 3, 6, 7	1, 2, 3, 4, 7	1
1, 2, 3, 4, 7	1, 2, 3, 4, 5	1
1, 2, 3, 4, 5	2, 3, 4, 5, 6	0
2, 3, 4, 5, 6	3, 4, 5, 6, 7	0
3, 4, 5, 6, 7	1, 2, 4, 5, 6, 7	1
1, 2, 4, 5, 6, 7	1, 2, 3, 5, 6, 7	1
1, 2, 3, 5, 6, 7	1, 2, 3, 4, 6, 7	1
1, 2, 3, 4, 6, 7	1, 2, 3, 4, 5, 7	1
1, 2, 3, 4, 5, 7	1, 2, 3, 4, 5, 6	1
1, 2, 3, 4, 5, 6	2, 3, 4, 5, 6, 7	0
2, 3, 4, 5, 6, 7	1, 2, 3, 4, 5, 6 7	1
1, 2, 3, 4, 5, 6, 7	1, 2, 3, 4, 5, 6, 7	1

(b)

	a	b	c	
$\to 1$	2	2	2	1
2	3	1	1, 2	1
3	4	3	$\emptyset$	1
1, 2	2, 3	1, 2	1, 2	1
4	5	4	4	1
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	0
2, 3	3, 4	1, 3	1, 2	1
5	1	5	5	1
3, 4	4, 5	3, 4	4	1
1, 3	2, 4	2, 3	2	1
4, 5	1, 5	4, 5	4, 5	1
2, 4	3, 5	1, 4	1, 2, 4	1
1, 5	1, 2	2, 5	2, 5	1
3, 5	1, 4	3, 5	5	1
1, 4	2, 5	2, 4	2, 4	1
1, 2, 4	2, 3, 5	1, 2, 4	1, 2, 4	1
2, 5	1, 3	1, 5	1, 2, 5	1
2, 3, 5	1, 3, 4	1, 3, 5	1, 2, 5	1
1, 2, 5	1, 2, 3	1, 2, 5	1, 2, 5	1
1, 3, 4	2, 4, 5	2, 3, 4	2, 4	1
1, 3, 5	1, 2, 4	2, 3, 5	2, 5	1
1, 2, 3	2, 3, 4	1, 2, 3	1, 2	1
2, 4, 5	1, 3, 5	1, 4, 5	1, 2, 4, 5	1
2, 3, 4	3, 4, 5	1, 3, 4	1, 2, 4	1
1, 4, 5	1, 2, 5	2, 4, 5	2, 4, 5	1
1, 2, 4, 5	1, 2, 3, 5	1, 2, 4, 5	1, 2, 4, 5	1
3, 4, 5	1, 4, 5	3, 4, 5	4, 5	1
1, 2, 3, 5	1, 2, 3, 4	1, 2, 3, 5	1, 2, 5	1
1, 2, 3, 4	2, 3, 4, 5	1, 2, 3, 4	1, 2, 4	1
2, 3, 4, 5	1, 3, 4, 5	1, 3, 4, 5	1, 2, 4, 5	1
1, 3, 4, 5	1, 2, 4, 5	2, 3, 4, 5	2, 4, 5	1

(c)

	a	b	c	
$\to 1$	2	2	2	1
2	3	1	2, 3	1
3	4	3	$\emptyset$	1
2, 3	3, 4	1, 3	2, 3	1
4	5	4	4	1
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	0
3, 4	4, 5	3, 4	4	1
1, 3	2, 4	2, 3	2	1
5	1	5	5	1
4, 5	1, 5	4, 5	4, 5	1
2, 4	3, 5	1, 4	2, 3	1
1, 5	6	2, 5	2, 5	1
3, 5	1, 4	3, 5	5	1
1, 4	2, 5	2, 4	2, 4	1
6	2, 3	1, 2	2, 3	1
2, 5	1, 3	1, 5	2, 3, 5	1
2, 3, 5	1, 3, 4	1, 3, 5	2, 3, 5	1
1, 3, 4	2, 4, 5	2, 3, 4	2, 4	1
1, 4, 5	1, 2, 4	2, 4, 5	2, 4, 5	1
2, 4, 5	1, 3, 5	1, 4, 5	2, 3, 4, 5	1
2, 3, 4	3, 4, 5	1, 3, 4	2, 3, 4	1
1, 2, 4	2, 3, 5	1, 2, 4	2, 3, 4	1
1, 3, 5	1, 2, 4	2, 3, 5	2, 5	1
2, 3, 4, 5	1, 3, 4, 5	1, 3, 4, 5	2, 3, 4, 5	1
3, 4, 5	1, 4, 5	3, 4, 5	4, 5	1
1, 3, 4, 5	1, 2, 4, 5	2, 3, 4, 5	2, 4, 5	1
1, 2, 4, 5	1, 2, 3, 5	1, 2, 4, 5	2, 3, 4, 5	1
1, 2, 3, 5	1, 2, 3, 4	1, 2, 3, 5	2, 3, 5	1
1, 2, 3, 4	2, 3, 4, 5	1, 2, 3, 4	2, 3, 4	1

□

Exercise 2: Reduce the following DFAs:

(a)

	a	b	
$\to 1$	2	3	0
2	3	2	1
3	4	5	0
4	1	8	1
5	6	7	0
6	7	6	1
7	8	1	0
8	5	4	1

(b)

	a	b	
$\to 1$	2	3	0
2	3	2	1
3	4	5	0
4	1	8	1
5	6	7	0
6	7	6	1
7	8	1	0
8	5	5	1

(c) Your result of 1(b).

(d) Your result of 1(c).

Solution.

(a)

Rejecting	Accepting
1, 3, 5, 7	2, 4, 6, 8
1, 3, 5, 7	2, 4, 6, 8

Setting $p = \{1, 3, 5, 7\}$ and $q = \{2, 4, 6, 8\}$:

	a	b	
$\to p$	q	p	0
q	p	q	1

(b)

<table>
<tr><td colspan="4">Rejecting</td><td colspan="4">Accepting</td></tr>
<tr><td colspan="4">1, 3, 5, 7</td><td colspan="4">2, 4, 6, 8</td></tr>
<tr><td colspan="4">1, 3, 5, 7</td><td colspan="3">2, 4, 6</td><td>8</td></tr>
<tr><td colspan="3">1, 3, 5</td><td>7</td><td colspan="2">2, 6</td><td>4</td><td>8</td></tr>
<tr><td>1</td><td>3</td><td>5</td><td>7</td><td>2</td><td>6</td><td>4</td><td>8</td></tr>
</table>

The DFA is already reduced.

□

Exercise 3: Construct NFAs for the following regular expressions using the construction given in class; then find the corresponding DFAs; then reduce them:

(a) $(a^2 \cup a^3 \cup a^5)^*$ over $\{a\}$

(b) $(a^2)^*(a^3)^*(a^5)^*$ over $\{a\}$

(c) $(abc \cup ab)^* aa^*(ab)^*$ over $\{a, b, c\}$

(d) $0^*(00 \cup 11)^*(01 \cup 10)^* 1^*$ over $\{0, 1\}$

Solution.

(a) NFA for a

	a	
$\to q_0$	q_1	0
q_1	$\emptyset$	1

NFA for a

	a	
$\to q_0$	q_2	0
q_2	$\emptyset$	1

Concatenate these to get a^2

NFA for a^2

	a	
$\to q_0$	q_1	0
q_1	q_2	0
q_2	$\emptyset$	1

Similarly, we have

NFA for a^3

	a	
$\to q_0$	q_3	0
q_3	q_4	0
q_4	q_5	0
q_5	$\emptyset$	1

NFA for a^5

	a	
$\to q_0$	q_6	0
q_6	q_7	0
q_7	q_8	0
q_8	q_9	0
q_9	$\emptyset$	1

(b)

□

Exercise 4: Construct regular expressions for the languages accepted by the following automata:

(a)

	a	b	c	
$\to 1$	2	2	2	1
2	3	1	2, 3	1
3	4	3	$\emptyset$	1
4	1	4	4	1

(b)

	a	b	
$\to A$	B	C	0
B	A	C	0
C	B	A	1

Solution. (a)

$$\begin{aligned}
X_1 &= aX_2 \cup bX_2 \cup cX_2 \cup \varepsilon \\
X_2 &= aX_3 \cup bX_1 \cup c(X_2 \cup X_3) \cup \varepsilon \\
X_3 &= aX_4 \cup bX_3 \cup \varepsilon \\
X_4 &= aX_1 \cup bX_4 \cup cX_4 \cup \varepsilon
\end{aligned}$$

Solving for X_4

$$\begin{aligned}
X_4 &= aX_1 \cup bX_4 \cup cX_4 \cup \varepsilon \\
&= aX_1 \cup (b \cup c)X_4 \cup \varepsilon \\
&= (b \cup c)X_4 \cup aX_1 \cup \varepsilon \\
&= (b \cup c)^*(aX_1 \cup \varepsilon) \\
&= (b \cup c)^* aX_1 \cup (b \cup c)^*
\end{aligned}$$

Similarly,

$$\begin{aligned}
X_3 &= aX_4 \cup bX_3 \cup \varepsilon \\
&= a\left((b \cup c)^* aX_1 \cup (b \cup c)^*\right) \cup bX_3 \cup \varepsilon \\
&= a(b \cup c)^* aX_1 \cup a(b \cup c)^* \cup bX_3 \cup \varepsilon \\
&= bX_3 \cup a(b \cup c)^* aX_1 \cup a(b \cup c)^* \cup \varepsilon \\
&= b^*(a(b \cup c)^* aX_1 \cup a(b \cup c)^* \cup \varepsilon) \\
&= b^* a(b \cup c)^* aX_1 \cup b^* a(b \cup c)^* \cup b^*
\end{aligned}$$

Solving X_2:

$$\begin{aligned}
X_2 &= aX_3 \cup bX_1 \cup c(X_2 \cup X_3) \cup \varepsilon \\
&= (a \cup c)X_3 \cup bX_1 \cup cX_2 \cup \varepsilon \\
&= (a \cup c)(b^* a(b \cup c)^* aX_1 \cup b^* a(b \cup c)^* \cup b^*) \cup bX_1 \cup cX_2 \cup \varepsilon \\
&= cX_2 \cup (a \cup c)(b^* a(b \cup c)^* aX_1 \cup b^* a(b \cup c)^* \cup b^*) \cup bX_1 \cup \varepsilon \\
&= c^*((a \cup c)(b^* a(b \cup c)^* aX_1 \cup b^* a(b \cup c)^* \cup b^*) \cup bX_1 \cup \varepsilon) \\
&= c^*(a \cup c)(b^* a(b \cup c)^* aX_1 \cup c^* b^* a(b \cup c)^* \cup c^* b^*) \cup c^* bX_1 \cup c^* \\
&= c^*(a \cup c)b^* a(b \cup c)^* aX_1 \cup c^*(a \cup c)(c^* b^* a(b \cup c)^* \cup c^* b^*) \cup c^* bX_1 \cup c^* \\
&= (c^*(a \cup c)b^* a(b \cup c)^* a \cup c^* b)X_1 \cup c^*(a \cup c)(c^* b^* a(b \cup c)^* \cup c^* b^*) \cup c^*
\end{aligned}$$

Finally, solving for X_1:

$$\begin{aligned}
X_1 &= aX_2 \cup bX_2 \cup cX_2 \cup \varepsilon \\
&= (a \cup b \cup c)X_2 \cup \varepsilon \\
&= (a \cup b \cup c)((c^*(a \cup c)b^* a(b \cup c)^* a \cup c^* b)X_1 \cup c^*(a \cup c)(c^* b^* a(b \cup c)^* \cup c^* b^*) \cup c^*) \cup \varepsilon \\
&= (a \cup b \cup c)(c^*(a \cup c)b^* a(b \cup c)^* a \cup c^* b)X_1 \cup (a \cup b \cup c)c^*(a \cup c)(c^* b^* a(b \cup c)^* \cup c^* b^*) \cup c^* \cup \varepsilon \\
&= \left((a \cup b \cup c)(c^*(a \cup c)b^* a(b \cup c)^* a \cup c^* b)\right)^*((a \cup b \cup c)c^*(a \cup c)(c^* b^* a(b \cup c)^* \cup c^* b^*) \cup c^*) \cup \varepsilon
\end{aligned}$$

(b)

$$\begin{aligned}
X_A &= aX_B \cup bX_C \\
X_B &= aX_A \cup bX_C \\
X_C &= aX_B \cup bX_A \cup \varepsilon
\end{aligned}$$

Plugging in the equation for X_C into X_B

$$\begin{aligned}
X_B &= aX_A \cup b(aX_B \cup bX_A \cup \varepsilon) \\
&= aX_A \cup baX_B \cup b^2 X_A \cup b \\
&= baX_B \cup (ba \cup b^2)X_A \cup b \\
&= (ba)^*((ba \cup b^2)X_A \cup b) \\
&= (ba)^*(ba \cup b^2)X_A \cup (ba)^* b
\end{aligned}$$

We substitute back into X_C:

$$\begin{aligned} X_C &= aX_B \cup bX_A \cup \varepsilon \\ &= a((ba)^*(ba \cup b^2)X_A \cup (ba)^*b) \cup bX_A \cup \varepsilon \end{aligned}$$

We now substitute the new equations for X_B and X_C into the equation for X_A

$$\begin{aligned} X_A &= aX_B \cup bX_C \\ &= a((ba)^*(ba \cup b^2)X_A \cup (ba)^*b) \cup b(a((ba)^*(ba \cup b^2)X_A \cup (ba)^*b) \cup bX_A \cup \varepsilon) \\ &= a(ba)^*(ba \cup b^2)X_A \cup a(ba)^*b \cup ba((ba)^*(ba \cup b^2)X_A \cup b(ba)^*b) \cup b^2X_A \cup b \\ &= (a(ba)^*(ba \cup b^2) \cup ba(ba)^*(ba \cup b^2) \cup b^2)X_A \cup bab(ba)^*b \cup b^2X_A \cup b \\ &= (a(ba)^*(ba \cup b^2) \cup ba(ba)^*(ba \cup b^2) \cup b^2)^* bab(ba)^*b \cup b \end{aligned}$$

□

2.2 Exercise Set 2

Exercise 1: Prove that the following languages are not regular:

(a) $L = \{x \in (0 \cup 1)^*2(0 \cup 1)^* \mid \text{number of 0s before 2} = \text{number of 1s after 2}\}$

(b) $L = \{x \in (0 \cup 1)^*2(0 \cup 1)^* \mid \text{number of 0s before 2} \neq \text{number of 1s after 2}\}$

(c) $L = \{a^{i^2} \mid i \geq 1\}$

(d) $L = \{a^{2^i} \mid i \geq 1\}$

Solution.

(a) Suppose, by way of contradiction, that L is regular and that p is its pumping length. Consider the string $s = 0^p21^p$. Clearly, $|s| \geq p$. Thus, by the **Pumping Lemma**, there exist strings s_1, s_2, s_3 such that $s = s_1s_2s_3$ with $|s_1s_2| \leq p$ and $|s_2| \geq 1$ and, for all $n \geq 0$, $s_1s_2^ns_3 \in L$. Observe that $s_1s_2 = 0^k$ for some $k \leq p$ (for otherwise $|s_1s_2| > p$), hence $s_3 = 0^{p-k}21^p$. Thus, we write $s_1 = 0^{k-q}$ and $s_2 = 0^q$ for some $q \geq 1$. By the pumping lemma,

$$\begin{aligned} s_1s_2^ns_3 &= 0^{k-q}(0^q)^n0^{p-k}21^p \\ &= 0^{k-q}0^{qn}0^{p-k}21^p \\ &= 0^{p+q(n-1)}21^p \end{aligned}$$

is in L. However, for $n \geq 2$, there are more 0s before the 2 than 1s after, hence $s_1s_2^ns_3 \notin L$. A contradiction. Thus, L is not regular.

(b) Suppose, by way of contradiction, that L is regular and that p is its pumping length. Consider the string $s = 0^p21^{p+p!}$. Clearly, $|s| \geq p$. Thus, by the **Pumping Lemma**, there exist strings s_1, s_2, s_3 such that $s = s_1s_2s_3$ with $|s_1s_2| \leq p$ and $|s_2| \geq 1$ and, for all $n \geq 0$, $s_1s_2^ns_3 \in L$. Observe that $s_1s_2 = 0^k$ for some $k \leq p$ (for otherwise $|s_1s_2| > p$), hence $s_3 = 0^{p-k}21^{p+p!}$. Thus, we write $s_1 = 0^{k-q}$ and $s_2 = 0^q$ for some $q \geq 1$. By the pumping lemma,

$$\begin{aligned} s_1s_2^ns_3 &= 0^{k-q}(0^q)^n0^{p-k}21^{p+p!} \\ &= 0^{k-q}0^{qn}0^{p-k}21^{p+p!} \\ &= 0^{p+q(n-1)}21^{p+p!} \end{aligned}$$

is in L. Now, since $q \leq p$, $q \mid p!$. Thus, taking $n = \frac{p!}{q} + 1$, we have

$$\begin{aligned} s_1s_2^ns_3 &= 0^{p+q(\frac{p!}{q}+1-1)}21^{p+p!} \\ &= 0^{p+q(\frac{p!}{q})}21^{p+p!} \\ &= 0^{p+p!}21^{p+p!} \end{aligned}$$

Thus, $s_1s_2^ns_3 \notin L$, a contradiction. Therefore L is not regular.

(c) In order to reach a contradiction, suppose L is regular and that p is its pumping length. Consider the string $s = a^{p^2}$. By the pumping lemma, we have $s = s_1 s_2 s_3$. Since $|s_1 s_2| \leq p$, this forces $s_1 s_2 = a^k$ for some $0 < k \leq p$ and $s_3 = a^{p^2-k}$. Then $s_1 = a^{k-r}$ and $s_2 = a^r$ for some $0 < r \leq k$. Then

$$\begin{aligned} s_1 s_2^n s_3 &= a^{k-r}(a^r)^n a^{p^2-k} \\ &= a^{k-r} a^{rn} a^{p^2-k} \\ &= a^{p^2+rn-r} \\ &= a^{p^2+r(n-1)} \end{aligned}$$

Take $n = 2$. Then $s_1 s_2^n s_3 = a^{p^2+r} \in L$. However, $p^2 + r$ cannot be a perfect square: since $r \leq p$, we have

$$\begin{aligned} p^2 + r &\leq p^2 + p \\ &< p^2 + p + 1 \\ &= (p+1)^2 \end{aligned}$$

Thus, $s_1 s_2^n s_3 \notin L$, a contradiction. Therefore L is not regular.

(d) In order to reach a contradiction, suppose L is regular and that p is its pumping length. Consider the string $s = a^{2^p}$. By the pumping lemma, we have $s = s_1 s_2 s_3$. Since $|s_1 s_2| \leq p$, this forces $s_1 s_2 = a^k$ for some $0 < k \leq p$ and $s_3 = a^{2^p-k}$. Then $s_1 = a^{k-r}$ and $s_2 = a^r$ for some $0 < r \leq k$. Then

$$\begin{aligned} s_1 s_2^n s_3 &= a^{k-r}(a^r)^n a^{2^p-k} \\ &= a^{k-r} a^{rn} a^{2^p-k} \\ &= a^{2^p+rn-r} \\ &= a^{2^p+r(n-1)} \end{aligned}$$

Take $n = 2$. Then $s_1 s_2^n s_3 = a^{2^p+r} \in L$. However, $2^p + r$ cannot be a power of 2: since $r \leq p$, we have

$$\begin{aligned} 2^p + r &\leq 2^p + p \\ &< 2^p + 2^p \\ &= 2^{p+1} \end{aligned}$$

Thus, $s_1 s_2^n s_3 \notin L$, a contradiction. Therefore L is not regular.

$\square$

Exercise 2: Construct DFAs for the following extended regular expressions:

(a) $\left[\overline{(000)^*} \cap \left((01)^* \cup (10)^*\right)\right] \cap \overline{(11)^*}$ over $\{0, 1\}$

(b) $\overline{(0 \cup 1)^* 01^* 10^* 01^* 1 (0 \cup 1)^*} - (10 \cup 01)^*$ over $\{0, 1, 2\}$

Exercise 3: Construct Chomsky Normal Form grammars for $L(G) - \varepsilon$ for the following cfgs G:

(a) $G = (\{a, b\}, \{S, A\}, S, \{S \to aAAaS \mid a, A \to bAA \mid aSSSA \mid \varepsilon\})$

(b) $G = (\{a, b, c\}, \{S, A, B\}, S, \{S \to aAA \mid A, A \to bBBB \mid B \mid \varepsilon, B \to bSSS \mid S \mid \varepsilon\})$

Solution.

(a)

$$\begin{aligned}S &\to aAAaS \mid a\\ A &\to bAA \mid aSSSA \mid \varepsilon\end{aligned}$$

There are no useless productions, since $S \Rightarrow a$ and $A \Rightarrow b$. A is clearly nullable, so we get

$$\begin{aligned}S &\to aAAaS \mid a\cancel{A}AaS \mid aA\cancel{A}aS \mid a\cancel{A}\cancel{A}aS \mid a\\ A &\to bAA \mid b\cancel{A}A \mid bA\cancel{A} \mid b\cancel{A}\cancel{A} \mid aSSSA \mid aSSS\cancel{A}\end{aligned}$$

which reduces to

$$\begin{aligned}S &\to aAAaS \mid aAaS \mid aaS \mid a\\ A &\to bAA \mid bA \mid b \mid aSSSA \mid aSSS\end{aligned}$$

There are no unit productions. Setting $X_a \to a$ and $X_b \to b$, we have

$$\begin{aligned}S &\to X_aAAX_aS \mid X_aAX_aS \mid X_aX_aS \mid a\\ A &\to X_bAA \mid X_bA \mid b \mid X_aSSSA \mid X_aSSS\end{aligned}$$

Finally, we decompose:

$$\begin{aligned}S &\to X_aS_1 \mid X_aS_2 \mid X_aS_3 \mid a\\ A &\to X_bA_1 \mid X_bA \mid b \mid X_aA_2 \mid X_aA_5\\ S_1 &\to AS_2\\ S_2 &\to AS_3\\ S_3 &\to X_aS\\ A_1 &\to AA\\ A_2 &\to SA_3\\ A_3 &\to SA_4\\ A_4 &\to SA\\ A_5 &\to SA_6\\ A_6 &\to SS\end{aligned}$$

(b)

$$\begin{aligned}S &\to aAA \mid A\\ A &\to bBBB \mid B \mid \varepsilon\\ B &\to bSSS \mid S \mid \varepsilon\end{aligned}$$

There are no useless productions, since $S \Rightarrow a$, $A \Rightarrow b$ and $B \Rightarrow b$. Clearly, S, A, and B are all nullable. Thus, we have

$$\begin{aligned}S &\to aAA \mid a\cancel{A} \mid aA\cancel{A} \mid a\cancel{A}\cancel{A} \mid A \mid \cancel{A}\\ A &\to bBBB \mid b\cancel{B}BB \mid bB\cancel{B}B \mid bBB\cancel{B} \mid b\cancel{B}\cancel{B}B \mid b\cancel{B}B\cancel{B} \mid bB\cancel{B}\cancel{B} \mid B \mid \cancel{B}\\ B &\to bSSS \mid b\cancel{S}SS \mid bS\cancel{S}S \mid bSS\cancel{S} \mid b\cancel{S}\cancel{S}S \mid b\cancel{S}S\cancel{S} \mid bS\cancel{S}\cancel{S} \mid b\cancel{S}\cancel{S}\cancel{S} \mid S \mid \cancel{S}\end{aligned}$$

which reduces to

$$
\begin{aligned}
S &\to aAA \mid a \mid aA \mid A \\
A &\to bBBB \mid bBB \mid bB \mid B \\
B &\to bSSS \mid bSS \mid bS \mid b \mid S
\end{aligned}
$$

We eliminate the unit productions $S \to A$, $A \to B$ and $B \to S$.

$$
\begin{aligned}
S &\to aAA \mid a \mid aA \mid bBBB \mid bBB \mid bB \\
A &\to bBBB \mid bBB \mid bB \mid bSSS \mid bSS \mid bS \mid b \\
B &\to bSSS \mid bSS \mid bS \mid b \mid aAA \mid a \mid aA \mid bBBB \mid bBB \mid bB
\end{aligned}
$$

We now set $X_a \to a$ and $X_b \to b$:

$$
\begin{aligned}
S &\to X_aAA \mid a \mid X_aA \mid X_bBBB \mid X_bBB \mid X_bB \\
A &\to X_bBBB \mid X_bBB \mid X_bB \mid X_bSSS \mid X_bSS \mid X_bS \mid b \\
B &\to X_bSSS \mid X_bSS \mid X_bS \mid b \mid X_aAA \mid a \mid X_aA \mid X_bBBB \mid X_bBB \mid X_bB
\end{aligned}
$$

Finally, we decompose:

$$
\begin{aligned}
S &\to X_aS_1 \mid a \mid X_aA \mid X_bS_2 \mid X_bS_3 \mid X_bB \\
A &\to X_bS_2 \mid X_bS_3 \mid X_bB \mid X_bA_1 \mid X_bA_2 \mid X_bS \mid b \\
B &\to X_bA_1 \mid X_bA_2 \mid X_bS \mid b \mid X_aS_1 \mid a \mid X_aA \mid X_bS_2 \mid X_bS_3 \mid X_bB \\
S_1 &\to AA \\
S_2 &\to BS_3 \\
S_3 &\to BB \\
A_1 &\to SA_2 \\
A_2 &\to SS
\end{aligned}
$$

□

Exercise 4: Construct Greibach Normal Form grammars for $L(G) - \varepsilon$ for the following cfgs G:

(a) $G = (\{a, b\}, \{S, A, B\}, S, \{S \to SaS \mid A, A \to AAAb \mid B \mid \varepsilon, B \to SSS \mid a\})$

(b) $G = (\{a, b\}, \{S, A, B, C\}, S, \{S \to ASS \mid a, A \to bBBB \mid BAA \mid \varepsilon, B \to CSS \mid SSC, C \to SS \mid b\})$

Solution.

(a)

(b)

□

2.3 Exercise Set 3

Exercise 1: Prove that the following languages are not context-free:

(a) $L = \{0^i1^j2^k \mid 0 \leq i < j < k\}$

(b) $L = \{0^{n^2}1^n \mid n \geq 0\}$

(c) $L = \{0^n1^n2^n \mid n \geq 0\}$

(d) $L = \{0^i1^j2^k \mid i > 2j > 3k \geq 1\}$

Solution.

(a) Suppose, by way of contradiction, that L is context-free with pumping length p. Then, by the pumping lemma, there exist strings s_1, s_2, s_3, s_4, and s_5 such that

(b) In order to reach a contradiction, suppose that L is context-free with pumping length p. Then, by the pumping lemma, there exist strings s_1, s_2, s_3, s_4, and s_5 such that

$$s = 0^{p^2}1^p = s_1s_2s_3s_4s_5$$

with $|s_2s_3s_4| \leq p$ and $|s_2s_4| \geq 1$. There are five cases, corresponding to the first occurrence of a 1 in s

Case 1: If the first 1 occurs in s_1, then we write $s_1 = 0^{p^2}1^i$, $s_2 = 1^j$, $s_3 = 1^k$, $s_4 = 1^l$, and $s_5 = 1^{p-(i+j+k+l)}$. Then

$$\begin{aligned} s_1s_2^ns_3s_4^ns_5 &= 0^{p^2}1^i1^{nj}1^k1^{nl}1^{p-(i+j+k+l)} \\ &= 0^{p^2}1^{p+(n-1)(j+l)} \end{aligned}$$

Taking $n = 2$ yields a string with p^2 0s but $p + j + l > p$ 1s, a contradiction.

Case 2: If the first 1 occurs in s_2, then we write $s_1 = 0^i$, $s_2 = 0^{p^2-i}1^j$, $s_3 = 1^k$, $s_4 = 1^l$, and $s_5 = 1^{p-(j+k+l)}$. Then

$$\begin{aligned} s_1s_2^ns_3s_4^ns_5 &= 0^i0^{n(p^2-i)}1^{nj}1^k1^{nl}1^{p-(j+k+l)} \\ &= 0^{np^2-(n-1)i}1^{p+(n-1)(j+l)} \end{aligned}$$

Taking $n = 0$ yields a string

□

Exercise 2: Construct PDAs P such that

i. $L = L(P)$

ii. $L = N(P)$

for the following languages:

(a) $L = \{0^i1^i \mid i \geq 0\} \cup \{0^i1^{2i} \mid i \geq 0\}$

(b) $L = L(G)$ with G given by $S \to aSd \mid aAd, A \to bAc \mid bc$

(c) $L = L(G)$ with G given by $S \to aSd \mid aAd, A \to bAc \mid bSc \mid \varepsilon$

(d) $L = L(G)$ with G given by $E \to +EE \mid *EE \mid id$

Exercise 3: Construct a grammar for the language $N(P)$:

$$P = (\{p, q\}, \{0, 1\}, \{Z, X\}, \delta, p, Z, \emptyset)$$

where

$$\begin{array}{ccc} \delta(p, 1, Z) = \{(p, XZ)\} & \delta(p, \varepsilon, Z) = \{(p, \varepsilon)\} & \delta(p, 1, X) = \{(p, XX)\} \\ \delta(a, 1, X) = \{(q, \varepsilon)\} & \delta(p, 0, X) = \{(q, X)\} & \delta(q, 0, Z) = \{(p, Z)\} \end{array}$$

Exercise 4: Construct Turing Machines for each of the languages in exercise 1.

Chapter 3

Exam Cheatsheets

3.1 Exam 1

3.1.1 Converting an NFA to a DFA

Begin with the initial state. Apply the transitions and add any newly visited states. Stop when no new states can be visited.

3.1.2 Reducing a DFA

Partition all states into accepting vs rejecting. Apply all transitions to a "representative" from a partition, then apply the transitions to the remaining members of the partition. If a member differs, move it to a new partition.

3.1.3 Converting a Regular Expression to an NFA

NFA for a:

	a	$b \neq a$	
$\rightarrow q_0$	q_1	$\emptyset$	0
q_1	$\emptyset$	$\emptyset$	1

NFA for ε:

	$c \in A$	
$\rightarrow q_0$	$\emptyset$	1

NFA for $\emptyset$:

	$c \in A$	
$\rightarrow q_0$	$\emptyset$	0

Union

Union initial states, copy remaining states from α and β. Final states are final states from α and β.

Concatenation

For final states of α, union the state from α with the initial state from β. Copy non-final states from α and non-initial states from β. Final states:

If the initial state is rejecting in β, final states are final states from β.

If the initial state is accepting in β, final states are final states from α and β, except the initial state in β.

Kleene Closure

Union final states with the initial state. Copy non-final states. Final states are final states from α and the initial state.

3.1.4 Converting an NFA to a Regular Expression

Set up system of equations. Solve for equation corresponding to initial state. Remember the lemma: if

$$X = LX \cup M$$

then

$$X = L^* M$$

3.2 Exam 2

3.2.1 Proving that a Language is not Regular

Pick a word w in the language. Write $w = xyz$ and pump: show that $xy^n z$ is not in the language. This is a contradiction.

3.2.2 Eliminating ε Productions

Find the nullable non-terminals (variables). These are the non-terminals from which ε can be derived. Specifically, a variable A is nullable if it is of the form

$$A \to \varepsilon$$

or

$$A \to A_1 A_2 A_3 \dots A_n$$

where each A_i is nullable. Then, simply replace each *combination of nullable variables* with ε and eliminate ε from the right-hand side.

3.2.3 Finding a DFA for an Extended Regular Expression

This is the same as finding a DFA for a regular expression, except when finding the complement. To find the complement, turn the NFA into a DFA and flip the accepting and rejecting states. Recall De Morgan's Laws

$$\begin{aligned}
\overline{A \cup B} &= \overline{A} \cap \overline{B} \\
\overline{A \cap B} &= \overline{A} \cup \overline{B} \\
A - B &= A \cap \overline{B}
\end{aligned}$$

3.2.4 Converting to Chomsky Normal Form

Eliminate useless variables, ε-productions, and unit productions. Then replace non-solitary terminals a with X_a and add $X_a \to a$ to your grammar. Finally, decompose any $X \to X_1 X_2 \dots X_n$ as

$$\begin{aligned}
X &\to X_1 A_1 \\
A_1 &\to X_2 A_2 \\
A_2 &\to X_3 A_3 \\
&\vdots \\
A_{n-2} &\to X_{n-1} X_n
\end{aligned}$$

3.2.5 Converting to Greibach Normal Form

Eliminate useless variables, ε-productions, and unit productions. Order the variables. Then, where any higher-listed variable produces a lower-listed variable through left-recursion, replace the lower-listed variable. Remove all direct left-recursions with the following rule:

$$A \to A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \beta_3 \mid \ldots \mid \beta_n$$

where no β_i begins with A, by replacing the productions for A with

$$\begin{aligned} A &\to \beta_1 \mid \beta_2 \mid \beta_3 \mid \ldots \mid \beta_n \mid \beta_1 A' \mid \beta_2 A' \mid \beta_3 A' \mid \ldots \mid \beta_n A' \\ A' &\to \alpha_1 \mid \alpha_2 \mid \alpha_3 \mid \ldots \mid \alpha_m \mid \alpha_1 A' \mid \alpha_2 A' \mid \alpha_3 A' \mid \ldots \mid \alpha_m A' \end{aligned}$$

Finally, substitute productions until they are all of the form

$$X \to a$$

or

$$X \to aX_1X_2 \ldots X_n$$

3.3 Exam 3

3.3.1 Proving a Language is not Context-Free

In order to reach a contradiction, assume the language is context-free with pumping length p. Then, choose a string s in the language and write $s = uvwxy$. Pump s as $uv^n wx^n y$ and show that the pumped string is not in the language for some $n \geq 0$. Usually, this requires division into cases, e.g. "the string contains only the letter a".

3.3.2 Finding a Pushdown Automaton for a Language

For all PDAs, left is top of stack.

PDA for $0^i 1^{ni}$ accepting by Empty Stack

		0	1	ε
q_0	Z_0	$(q_0, Z^n Z_0)$	$\emptyset$	$\emptyset$
	Z	(q_0, Z^{n+1})	(q_1, Z)	$\emptyset$
q_1	Z_0	$\emptyset$	$\emptyset$	(q_1, ε)
	Z	$\emptyset$	(q_1, ε)	$\emptyset$

PDA for $0^{ni} 1^i$ accepting by Empty Stack

		0	1	ε
q_0	Z_0	(q_1, Z_0)	$\emptyset$	$\emptyset$
	Z	(q_1, Z)	(q_n, Z)	$\emptyset$
q_1	Z_0	(q_2, Z_0)	$\emptyset$	$\emptyset$
	Z	(q_2, Z)	$\emptyset$	$\emptyset$
		$\vdots$		
q_i	Z_0	(q_{i+1}, Z_0)	$\emptyset$	$\emptyset$
	Z	(q_{i+1}, Z)	$\emptyset$	$\emptyset$
		$\vdots$		
q_{n-1}	Z_0	(q_0, ZZ_0)	$\emptyset$	$\emptyset$
	Z	(q_0, Z^2)	$\emptyset$	$\emptyset$
q_n	Z_0	$\emptyset$	$\emptyset$	(q_n, ε)
	Z	$\emptyset$	(q_n, ε)	$\emptyset$

For $i \geq 0$, simply add (q_0, ε) to the ε column for q_0.

This PDA accepts by empty stack. To change to accepting by final state, add auxhiliary states q_0', q_f and stack letter Z_0':

PDA for 0^i1^{ni} accepting by Final State

		0	1	ε
q_0'	Z_0'	$\emptyset$	$\emptyset$	(q_0, Z_0Z_0')
	Z_0	$\emptyset$	$\emptyset$	$\emptyset$
	Z	$\emptyset$	$\emptyset$	$\emptyset$
q_0	Z_0'	$\emptyset$	$\emptyset$	$\emptyset$
	Z_0	(q_0, Z^nZ_0)	$\emptyset$	$\emptyset$
	Z	(q_0, Z^{n+1})	(q_1, Z)	$\emptyset$
q_1	Z_0'	$\emptyset$	$\emptyset$	(q_f, ε)
	Z_0	$\emptyset$	$\emptyset$	(q_1, ε)
	Z	$\emptyset$	(q_1, ε)	$\emptyset$
q_f	Z_0'			
	Z_0		accepting	
	Z			

PDA for $0^{ni}1^i$ accepting by Final State

		0	1	ε
q_0'	Z_0'	$\emptyset$	$\emptyset$	(q_0, Z_0Z_0')
	Z_0	$\emptyset$	$\emptyset$	$\emptyset$
	Z	$\emptyset$	$\emptyset$	$\emptyset$
q_0	Z_0'	$\emptyset$	$\emptyset$	$\emptyset$
	Z_0	(q_1, Z_0)	$\emptyset$	$\emptyset$
	Z	(q_1, Z)	(q_n, Z)	$\emptyset$
q_1	Z_0'	$\emptyset$	$\emptyset$	$\emptyset$
	Z_0	(q_2, Z_0)	$\emptyset$	$\emptyset$
	Z	(q_2, Z)	$\emptyset$	$\emptyset$
			$\vdots$	
q_i	Z_0'	$\emptyset$	$\emptyset$	$\emptyset$
	Z_0	(q_{i+1}, Z_0)	$\emptyset$	$\emptyset$
	Z	(q_{i+1}, Z)	$\emptyset$	$\emptyset$
			$\vdots$	
q_{n-1}	Z_0'	$\emptyset$	$\emptyset$	$\emptyset$
	Z_0	(q_0, ZZ_0)	$\emptyset$	$\emptyset$
	Z	(q_0, Z^2)	$\emptyset$	$\emptyset$
q_n	Z_0'	$\emptyset$	$\emptyset$	(q_f, ε)
	Z_0	$\emptyset$	$\emptyset$	(q_n, ε)
	Z	$\emptyset$	(q_n, ε)	$\emptyset$
q_f	Z_0'			
	Z_0		accepting	
	Z			

3.3.3 Converting a CFG into a PDA

First, convert the CFG into Greibach Normal Form. Then, the non-terminals (variables) are the rows (stack symbols) of your PDA, with a single state q. The columns are just the terminals. For any production $X \to aX_1X_2 \dots X_n$, add $(q, X_1X_2 \dots X_n)$ to the row, column (X, A). If $X \to a$, add (q, ε).

3.3.4 Converting a PDA into a CFG

Suppose there are n states $q_1, q_2, \ldots, q_n$. First, add the start productions:

$$S \to (q_1, Z, q_1) \mid (q_1, Z, q_2) \mid \ldots \mid (q_1, Z, q_n)$$

3.3.5 Constructing a Turing Machine