University of Houston

Notes

COSC 3340
Intro. to Automata and Computability

Ernst Leiss

Khalid Hourani

Contents

1 Formal Languages

1.1 Regular Languages

1.1.1 Introduction

Definition 1.1.1. An **Alphabet** is a finite, non-empty set of atomic symbols.

Definition 1.1.2. A **word** or **string** is any finite sequence of symbols from an alphabet.

Definition 1.1.3. The **length** of a string, s, denoted $|s|$, is the number of symbols in s.

Definition 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 1.1.5. 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. 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 1.1.6. Given an alphabet A, a **formal language** or simply **language** L is any subset of A^*.

Theorem 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 1.1.7. Given two languages L_1 and L_2, the concatenation $L_1 \cdot L_2$ is given by

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

Clearly, we have

$$L \cdot \emptyset = \emptyset = \emptyset \cdot L$$
$$L \cdot \{\varepsilon\} = L = \{\varepsilon\} \cdot L$$

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 1.1.8. 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 1.1.3. 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. □

1.1.2 Finite Automata

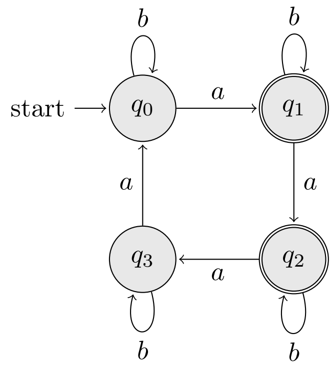
Definition 1.1.9. A **Deterministic Finite-State Automata** (DFA) or **Finite-State Machine** is a quintuple $(A, Q, \tau, q_0, \mathcal{F})$ where

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	
$\rightarrow q_0$	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 1.1.10. Let $\underset{\sim}{D}$ be some DFA. Then $L(\underset{\sim}{D})$, the language accepted by the DFA, is

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

Definition 1.1.11. A language is **regular** if and only if there exists a DFA that accepts it.

Definition 1.1.12. 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**

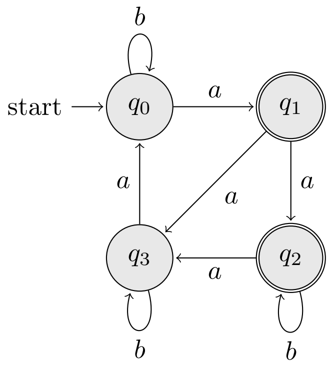
q_0 is the **initial state**

$\mathcal{F} \subseteq Q$ 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

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

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	
$\to q_0$	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
$\to q_0$	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
$\to q_0$	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
$\to q_0$	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
$\to q_0$	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
$\to q_0$	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
$\to q_0$	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
$\to q_0$	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	
$\to q_0$	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	
$\rightarrow 0$	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 1.1.13. 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

$$X_1 = aX_2 \cup bX_1 \cup bX_3 \tag{1}$$
$$X_2 = bX_3 \tag{2}$$
$$X_3 = aX_2 \cup aX_3 \cup bX_1 \cup \varepsilon \tag{3}$$

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

$$\begin{aligned} X_1 &= abX_3 \cup bX_1 \cup bX_3 \\ X_3 &= abX_3 \cup aX_3 \cup bX_1 \cup \varepsilon \end{aligned}$$

which we rewrite as

$$\begin{aligned} X_1 &= (ab \cup b)X_3 \cup bX_1 \\ X_3 &= (ab \cup a)X_3 \cup bX_1 \cup \varepsilon \end{aligned}$$

We now apply our lemma to the equation for X_3

$$\begin{aligned} X_1 &= (ab \cup b)X_3 \cup bX_1 \\ X_3 &= (ab \cup a)^*(bX_1 \cup \varepsilon) \end{aligned}$$

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	
$\to 1$	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 1.1.4. 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 1.1.14. 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 $\{0, 1\}$.

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

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

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

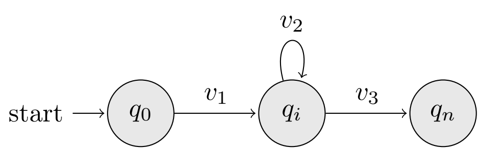
with $|Q| = n$. Consider the following set of equations

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

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 $s = a_1 a_2 \ldots a_n$. Let $v_1 = a_1 a_2 \ldots a_i$, $v_2 = a_{i+1} a_{i+2} \ldots a_j$, and $v_3 = a_{j+1} a_{j+2} \ldots a_n$. 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 1.1.5 (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.

1.2 Context-Free Languages

1.2.1 Context-Free Grammars

Definition 1.2.1. 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 1.2.2. Given strings α and β, we say α **derives** β if there exist $A, \alpha_1, \alpha_2, \gamma$ such that

$$\alpha = \alpha_1 A \alpha_2$$
$$\beta = \alpha_1 \gamma \alpha_2$$
$$A \to \gamma \in P$$

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

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

Definition 1.2.3. Given a context-free grammar G, the corresponding **context-free language** is

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

Theorem 1.2.1. Every regular language is a context-free language.

Proof. □

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

1.3 Normal Forms

1.3.1 Chomsky Normal Form

Definition 1.3.1. A context-free language is in **Chomsky Normal Form** if all of its productions are of the form

$$A \to BC$$

or

$$A \to a$$

or

$$S \to \varepsilon$$

Theorem 1.3.1. If G is a context-free grammar, there exists a Chomsky Normal Form grammar for $L(G) - \{\varepsilon\}$

1.3.2 Greibach Normal Form

Definition 1.3.2. A context-free language is in **Greibach Normal Form** if all of its productions are of the form

$$A \to aA_1A_2 \ldots A_n$$

or

$$S \to \varepsilon$$

For any CFG G, we preprocess the language:

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

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

$$S \to aSb|cAd$$
$$A \to aSc|bAd$$

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 $A \Rightarrow w$, then we can eliminate A.

Consider the following example:

$$S \to aS|bA|\varepsilon$$
$$A \to cAA|dBB$$
$$B \to aBA|bAA|cAC$$
$$C \to aCb|S$$

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

$$S \to aS|\varepsilon$$

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.

2 Exercise Sets

2.1 Exercise Set 1

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

(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

(b)

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

(b)

<table>
<tr><th colspan="4">Rejecting</th><th colspan="4">Accepting</th></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) $L = \{x \in (0 \cup 1)^* 2 (0 \cup 1)^* | \text{number of 0s before 2} = \text{number of 1s after 2}\}$

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

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

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

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 $n \geq 1$, $s_1s_2^ns_3 \in L$. 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 $n \geq 1$, $s_1s_2^ns_3 \in L$. 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 $q \leq p$, $q|p!$. 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.

□

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

NFA for ε:

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

NFA for $\emptyset$:

	$c \in A$	
$\to q_0$	$\emptyset$	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

$$X = L^* M$$