Regular Languages and Regular Expressions

Recall some of the notation from Note 1.

If \( L \subseteq A^* \) then the Kleene closure of \( L \) is the set \( L^* \) defined by

\[
L^* = \{ w_1w_2 \ldots w_n : n \geq 0 \text{ and } w_1, w_2, \ldots, w_n \in L \}
\]

**Notation.** \( \{a\}^* = \{ \epsilon, a, aa, aaaa, \ldots \} \) and \( \{a\}^+ = \{ a, aa, aaaa, \ldots \} \)

If we define \( L^n \) by \( L^0 = \{ \epsilon \} \), \( L^{n+1} = L^n L \) for \( n \geq 0 \), then \( L^* = \bigcup_{n=0}^{\infty} L^n \).

**Examples.** Let \( A = \{a, b, c, d\} \). Then

\[
\begin{align*}
\{a\}\{b\} & = \{ab\} \\
\{a, b\}\{c, d\} & = \{ac, ad, bc, bd\} \\
\{a\}^* & = \{ \epsilon, a, a^2, a^3, \ldots \} \\
\{c, d\}^* & = \{ \epsilon, c, d, c^2, d^2, cd, dc, c^3, d^3, \ldots \} \\
\{a\}^+\{c, d\} & = \{c, d, ac, ad, a^2c, a^2d, \ldots \}
\end{align*}
\]

A regular expression is an expression constructed from union, concatenation (a.k.a. product) and closure, for example

\[
\{a\}^* \cup \{b\}^* = \{ \epsilon, a, b, aa, bb, a^2, b^2, \ldots \}.
\]

A regular language is a language that can be represented using a regular expression. (We claim that palindromes are not a regular language. We will see why later.)

Let \( A \) be an alphabet. The class of regular languages (or regular sets) over \( A \) is defined recursively as follows:

1. any finite language is regular;
2. if \( L_1 \) and \( L_2 \) are regular then so is \( L_1 \cup L_2 \);
3. if \( L_1 \) and \( L_2 \) are regular then so is \( L_1L_2 \);
4. if \( L \) is regular then so is \( L^* \).

**Some more interesting examples.**

The set of strings over \( \{a, b\} \) that contain 3 \( a \)'s is represented by \( \{b\}^*a\{b\}^*a\{b\}^*a(b)^* \).

The set of strings over \( \{a, b\} \) that contain 3 consecutive \( a \)'s is represented by \( \{a, b\}^*aaa\{a, b\}^* \).

**Observation.** Like for finite automata, there are many different regular expressions that can represent the same language. For example, the set of words over alphabet \( \{a, b\} \) of length at least 1 can be written as \( \{a, b\}\{a, b\}^* \) or as \( \{a, b\}^*\{a, b\} \) or as \( a\{a, b\}^* \cup b\{a, b\}^* \) etc...

**Kleene’s Theorem**

This important result says that regular expressions have the same expressive power as finite automata: we can translate any r.e. into a finite automaton that accepts its language, and we can take a finite automaton and convert it into an equivalent r.e. (Note: some languages are more conveniently expressed as FAs or r.e.'s, but we always can translate.)

**Kleene’s Theorem.** Let \( L \) be a language over an alphabet \( A \). Then \( L \) is regular if and only if it is the language accepted by some finite automaton with alphabet \( A \).

We start by showing that FAs accept regular languages.

**Proof that automata can be translated into r.e.'s.** We show that a language \( L \) accepted by an automaton is regular. Suppose \( L \) is accepted by the finite automaton \( A = (Q, A, \phi, i, T) \) which we may assume to be deterministic.

For each \( p, q \in Q \) we define \( L(p, q) \) to be the set of all words which label paths from \( p \) to \( q \) i.e.

\[
L(p, q) = \{ w \in A^* : \phi(p, w) = q \}.
\]

The language \( L \) consists of all words which label paths from the initial state \( i \) to a terminal state \( t \in T \). Considering each terminal state separately we have

\[
L = \bigcup_{t \in T} L(i, t).
\]

Since we know that a union of regular sets is regular, it is now sufficient to show that each \( L(i, t) \) is regular.

The idea behind the proof is to show that sets of the form \( L(p, q) \) are regular by induction on the number of states in paths from \( p \) to \( q \) (with labels in \( L(p, q) \)). To do this we need to extend our notation to keep a record of states in such paths.

We shall say that a path

\[
p_1 \rightarrow p_2 \rightarrow p_3 \rightarrow \cdots \rightarrow p_{n-1} \rightarrow p_n
\]

lies in a set \( R \) of states if \( \{p_1, p_2, \ldots, p_n\} \subseteq R \). The states \( p_2, p_3, \ldots, p_n \) will be called intermediate states of the path.

- If \( R \) is a set of states containing \( p \) and \( q \) then we define \( L(p, R, q) \) to be the set of words which label paths from \( p \) to \( q \) lying in \( R \).
If $V$ is a set of states containing neither $p$ nor $q$ then $Z(p, V, q)$ is defined to be the set of words which label paths from $p$ to $q$ whose intermediate states belong to $V$.

Finally we shall denote by $A(p, q)$ the set of all letters which cause a transition from $p$ to $q$ i.e.

$$A(p, q) = \{ a \in A : \phi(p, a) = q \}.$$  

Since $A$ is a finite set it follows that $A(p, q)$ is regular.

We show by induction that the following statement holds for $n > 1$.

Claim. Let $p, q$ be states and $R, V$ be sets of states satisfying $|R| < n$ and $|V| < n$ and with $p, q \in R$ and $p, q \notin V$. Then the sets $L(p, R, q)$ and $Z(p, V, q)$ are regular.

When $n = 2$ we have $|R| = 1$ so $p = q$ and $R = \{p\}$. Now $L(p, \{p\}, p)$ is the set of words which label paths that only ever visit state $p$, so $L(p, \{p\}, p) = A(p, p)^*$, which is regular. Also for $n = 2$ we have $|V| < 2$. If $|V| = 0$ then $Z(p, V, q) = Z(p, \emptyset, q) = A(p, q)$. If $|V| = 1$ then $V = \{r\}$ for some $r$ which is distinct from $p, q$. A path from $p$ to $q$ whose only intermediate state is $r$ is of the form

$$p \rightarrow r \rightarrow r \rightarrow \cdots \rightarrow r \rightarrow q$$

which shows that

$$Z(p, \{r\}, q) = A(p, r)A(r, r)^*A(r, q)$$

which is a regular set.

That completes the base cases of the induction. We next show that $L(p, R, q)$ is regular for $|R| = n$ using the above claim as inductive hypothesis, and then that $Z(p, V, q)$ is regular for $|V| = n$, using both the claim and the fact that $L(p, R, q)$ is regular for $|R| = n$.

A path whose label belongs to $L(p, R, q)$ can be decomposed into

1. a string of paths from $p$ to $p$ each of which contains no intermediate visit to $p$, followed by
2. a path from $p$ to $q$ which makes no intermediate visit to either $p$ or $q$, followed by
3. a path from $q$ to $q$ which makes no intermediate visit to $p$.

The labels of each of these subpaths belong respectively to the following sets:

1. $Z(p, R \setminus \{p\}, p)^*$,
2. $Z(p, R \setminus \{p, q\}, q)$,
3. $L(q, R \setminus \{p\}, q)$.

It follows that

$$L(p, R, q) = Z(p, R \setminus \{p\}, p)^*Z(p, R \setminus \{p, q\}, q)L(q, R \setminus \{p\}, q)$$

where the right-hand side is a regular set by the inductive hypothesis. Hence $L(p, R, q)$ is regular for all $R$ with $|R| = n$.

Next we show that $Z(p, V, q)$ is regular. Let $w \in Z(p, V, q)$. Then there is a path from $p$ to $q$ with label $w$ whose intermediate states all lie in $V$. Let $r$ be the state which immediately follows $p$ and let $s$ be the state which immediately precedes $q$. Then the path can be decomposed into

1. a path from $p$ to $r$ whose label is a single letter, followed by
2. a path from $r$ to $s$ which does not visit the states $p$ or $q$, followed by
3. a path from $s$ to $q$ whose label is a single letter.

Hence

$$w \in A(p, r)L(r, V, s)A(s, q).$$

By what we have shown above, $L(r, V, s)$ is regular and hence $A(p, r)L(r, V, s)A(s, q)$ is regular. Now $Z(p, V, q)$ consists of all such paths with different $r$ and $s$ so we have

$$Z(p, V, q) = \bigcup_{r,s \in V} A(p, r)L(r, V, s)A(s, q)$$

which is regular. This completes the inductive step. It follows that $L(i, t) = L(i, Q, t)$ is regular for every $t$, as required.

We have seen that deterministic and nondeterministic automata accept the same class of languages, so in future we can talk about languages accepted a finite automaton without specifying whether it is deterministic or non-deterministic.

**Proof that r.e.’s can be translated into automata.**

The first construction we look at takes two automata and joins them “in parallel”. The language accepted by the new machine is then the union of the languages accepted by the separate machines.

**Proposition 1.** Let $L_1$ and $L_2$ be languages over an alphabet $A$. If there are finite automata accepting $L_1$ and $L_2$ then there is a finite automaton accepting $L_1 \cup L_2$. 


Proof. Let $A_1 = (Q_1, A, \phi_1, i_1, T_1)$ be an automaton accepting $L_1$ and $A_2 = (Q_2, A, \phi_2, i_2, T_2)$ be an automaton accepting $L_2$, where we may assume both are deterministic. Define $A = (Q, A, \phi, i, T)$ as follows:

- $Q = Q_1 \times Q_2$
- $\phi((q_1, q_2), a) = (\phi_1(q_1, a), \phi_2(q_2, a))$ for all $q_1 \in Q_1$; $q_2 \in Q_2$; $a \in A$
- $i = (i_1, i_2)$.
- $T = \{(p, q) : p \in T_1 \text{ or } q \in T_2\}$

Then

$$\phi(i, w) = (\phi_1(i_1, w), \phi_2(i_2, w))$$

for all $w \in A^*$.

Now we have

$$w \in L_1 \cup L_2 \iff \phi(i, w) \in T_1 \text{ or } \phi(i, w) \in T_2$$

$$\iff \phi(i, w) = (p, q) \text{ where } p \in T_1 \text{ or } q \in T_2 \text{ (or both)}$$

So from the definition of the set $T$ of terminal states of $A$ we have

$$w \in L_1 \cup L_2 \iff \phi(i, w) \in T \iff w \text{ is accepted by } A.$$ 

Hence the language accepted by $A$ is $L_1 \cup L_2$. \hfill \square

As an application of this result we show that any finite language is accepted by some finite automaton.

Let $A = \{a_1, a_2, \ldots, a_n\}$ be an alphabet. We begin by giving constructions of finite automata that accept at most one word over $A$. The diagram below shows (deterministic) finite automata for the empty set, the set $\{\epsilon\}$, and the set $\{l_1 l_2 \ldots l_m\}$, where $l_1, \ldots, l_m \in A$.

![Finite Automaton Diagram]

Now assume by way of induction that every language containing $r$ words is accepted by some automaton. (We have seen that this holds for $r = 0$ or $r = 1$.) A language containing $r + 1$ words can be written as

$$\{w_1, w_2, \ldots, w_r, w_{r+1}\} = \{w_1, w_2, \ldots, w_r\} \cup \{w_{r+1}\}.$$ 

By proposition 1 the right-hand side is accepted by a finite automaton. Hence every finite language is accepted by some finite automaton.

The second construction looks at how two automata can be joined in series. This will give a machine that accepts the product of the languages accepted by the originals. Whenever $q$ is a state of the first machine for which $\phi(q, a)$ is a terminal state, we allow the combined machine a choice of moves — to the initial state of the second machine or to the state $\phi(q, a)$.

If the first machine enters a terminal state then it can either be accepting its input or merely “passing through” during a longer computation. We deal with this by allowing the combined machine a choice of transitions whenever the first machine is about to enter a terminal state. Any remaining input can continue to be read by the first machine or can be passed on to the second. This is done by making the combined machine nondeterministic.

Example. Let $A = \{a, b\}$. In the diagram below the top left automaton accepts the language $L_1 = \{a\} \{ab, a^2b\} \{a, a^2\}$ while the top right automaton accepts $L_2 = \{b^2\} A^*$. (To see that the first machine accepts $L_1$, observe that an accepting path must start with $a$, then contain a number of loops labelled with $ab$ or $a^2b$ before concluding with either $a^2$ or $a$.) Combining the two machines as shown, we obtain an automaton accepting the language $L_1 L_2 = \{a\} \{ab, a^2b\} \{a, a^2\} \{b^2\} A^*$.

![Combined Automaton Diagram]

We now prove the general result formally.

**Proposition 2.** Let $L_1$ and $L_2$ be languages over an alphabet $A$. If there are finite automata accepting $L_1$ and $L_2$ then there is a finite automaton accepting $L_1 L_2$. 

2-5
Proof. As before we can assume that $L_1$ and $L_2$ are accepted by deterministic automata $A_1 = (Q_1, A, \phi_1, i_1, T_1)$ and $A_2 = (Q_2, A, \phi_2, i_2, T_2)$. By renaming the states if necessary, we can assume that $Q_1 \cap Q_2 = \emptyset$. Define $A$ to be the automaton $(Q_1 \cup Q_2, A, \psi, i, T_2)$ where $\psi$ is defined for $q_1 \in Q_1$ by

$$
\psi(q_1, a) = \{ \phi_1(q_1, a) \} \quad \text{if} \quad \phi_1(q_1, a) \notin T_1,
$$

$$
\psi(q_1, a) = \{ \phi_1(q_1, a), i_2 \} \quad \text{if} \quad \phi_1(q_1, a) \in T_1.
$$

and for $q_2 \in Q_2$ by

$$
\psi(q_2, a) = \{ \phi_2(q_2, a) \}.
$$

We now have to verify that $L_1L_2$ is the language accepted by $A$. Suppose $w_1 \in L_1$ and $w_2 \in L_2$. Then $\phi_1(i_1, w_1) \in T_1$ and $\phi_2(i_2, w_2) \in T_2$. By the definition of $\psi$ we see that $i_2 \in \psi(i_1, w_1)$ and $\phi_2(i_2, w_2) \in \psi(i_1, w_1w_2)$. Now in $A$ the path with label $w_1w_2$ has the form

$$
i_1 \rightarrow w_1 \rightarrow \{ \ldots, i_2, \ldots \} \rightarrow w_2 \rightarrow \{ \ldots, \phi_2(i_2, w_2), \ldots \}
$$

so the final set $\psi(i_1, w_1w_2)$ contains a state from $T_2$, i.e. an accepting state of $A$. Hence $w_1w_2$ is accepted by $A$. This shows that $L_1L_2 \subseteq L(A)$.

Conversely, suppose that $w \in L(A)$. Then $\psi(i_1, w)$ contains a member of $T_2$. To find $\psi(i_1, w)$ we consider the path of the first machine $A_1$ with label $w$. If $t_1, t_2, \ldots, t_n$ are the accepting states of $A_1$ on this path then we have

$$
i_1 \rightarrow \ldots \rightarrow t_1 \rightarrow \ldots \rightarrow w_2 \rightarrow \ldots \rightarrow t_n \rightarrow \ldots \rightarrow \phi_1(i_1, w)
$$

where $w = w_1w_2 \ldots w_nw_{n+1}$ (if $w$ is accepted by $A_1$ then $w_{n+1} = \epsilon$ and $t_n = \phi_1(i_1, w)$). After the nondeterministic automaton $A$ has read the prefix $w_1w_2 \ldots w_i$, one possibility has it transforming to $i_2$ and then behaving like $A_2$ on the remaining input which leads to the final state $\phi_2(i_2, w_{i+1} \ldots w_{n+1})$. Thus the set $\psi(i_1, w)$ of all possible final states of $A$ is

$$\{ \phi_1(i_1, w), \phi_2(i_2, w_2 \ldots w_{n+1}), \phi_2(i_2, w_3 \ldots w_{n+1}), \ldots, \phi_2(i_2, w_nw_{n+1}) \}
$$

and since $w$ is accepted by $A$ one of these must belong to $T_2$. The first element (i.e. $\phi_1(i_1, w)$) belongs to $Q_1$, so we must have

$$\phi_2(i_2, w_{i+1} \ldots w_{n+1}) \in T_2 \quad \text{for some} \quad r
$$

i.e. $w_{r+1} \ldots w_{n+1} \in L_2$. But the input word $w_1 \ldots w_i$ transforms $A_1$ to the terminal state $t_i$ so $w_1 \ldots w_i \in L_1$. Now we have $w = (w_1 \ldots w_i)(w_{i+1} \ldots w_{n+1}) \in L_1L_2$. Hence $L(A) \subseteq L_1L_2$ and combining this with the other inclusion we proved gives $L(A) = L_1L_2$. So the new automaton we have constructed accepts $L_1L_2$.  

The third method we consider here for constructing new machines consists of linking the accepting states of an automaton back to its input. If $w_1, w_2, \ldots, w_n$ are words in the language $L$ accepted by the original machine, then in the new machine the accepting paths can be strung together to give a path which accepts the word $w_1w_2 \ldots w_n$. Thus we have an automaton which accepts the set

$$L^* = \{ w_1w_2 \ldots w_n : n \in \mathbb{N} \text{ and } w_1, w_2, \ldots, w_n \in L \}.$$

It then follows from proposition 1 that $A^* = A^+ \cup \{ \epsilon \}$ is acceptable by an automaton. In the general case some care needs to be taken with the construction if $L$ contains the empty string i.e. if the initial state is also an accepting state.

Example. Consider the two automata shown below. The automaton on the left accepts the language $\{a^2, b^2\}$ (shown on the right) is obtained by adding transitions which take accepting paths back to the initial state.

---

### Proposition 3.

If $L$ is a language accepted by a finite automaton then there is a finite automaton which accepts $L^*$.

**Proof.** This is similar to the proof of proposition 2. Here, whenever a terminal state is entered we allow the machine to simultaneously return to its initial state. The details are omitted. \qed