Kleene’s Theorem

Kleene’s Theorem says that a language is regular if and only if it is the language accepted by a DFA. But before we sketch a proof we need to introduce another class of finite state machines.

Non-deterministic finite automata

A DFA is deterministic in the sense that for each state and each input symbol, the next state is completely determined. We now consider a generalization of the idea of a deterministic finite automaton, namely a non-deterministic finite automaton or NFA for short.

In an NFA there can be several arrows with the same label starting at a state and there can be arrows labelled with the empty string $\varepsilon$. 
More precisely, an NFA consists of

- An alphabet $\Sigma$.

- A finite set $S$ of states.

- An initial state.

- A set $A \subseteq S$ of accepting states.

- A transition function $f : S \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(S)$.

This is almost the same description as for a DFA except that the transition function takes a state and an input symbol to a subset of the set of states.
It is perhaps easier to think of an NFA in terms of its diagram. The diagram is constructed in exactly the same way as for a DFA except that each state may have more than one arrow leaving it with the same label. Also, we are allowed to use the empty string \( \varepsilon \) to label arrows. Then \( f(Z, x) \) is the set of all states \( W \) such that there is an arrow from \( Z \) to \( W \) labelled \( x \).

**The language of an NFA**

Let \( M \) be an NFA. To each path \( p \) of arrows in the diagram of \( M \) we associate the string \( u(p) \) obtained by concatenating the symbols along \( p \). We say that a string \( v \) is **accepted** by \( M \) if there is some path \( p \) from the initial state to one of the accepting states such that \( u(p) = v \). The set of all strings accepted by \( M \) is the language \( L(M) \) accepted by \( M \).
It turns out that the language accepted by an NFA is regular and so by Kleene’s theorem there is always a DFA that accepts the same language. But if the NFA has \( m \) states then the DFA may have \( 2^m \) states.

On the other hand if we begin with a regular language it is often much easier to find an NFA that accepts it, rather than a DFA.

**Theorem** (Kleene, Rabin, Scott) *Given a finite alphabet \( \Sigma \), the following three statements are equivalent for a language \( L \subseteq \Sigma^* \)

1. \( L \) is recognized by a DFA
2. \( L \) is recognized by an NFA
3. \( L = L(r) \) for some regular expression \( r \)
Proof. Certainly (1) implies (2).

A proof that (1) implies (3) can be found on page 119 of the textbook.

This can be modified to give a proof that (2) implies (3) as follows:

Let $S$ be the set of states of an NFA $M$. We shall say that a path

$$A_1, u_1, A_2, u_2, \ldots, u_n, A_{n+1}$$

visits the states $A_1, \ldots, A_n$. Given states $A$, $B$ and a subset $X \subseteq S$ let $L(A, B, X)$ be the language consisting of the labels of all paths from $A$ to $B$ which visit only states in $X$. We shall prove by induction on $|X|$ that for all $X$ there exists a regular expression $\rho(A, B, X)$ which defines $L(A, B, X)$.

Suppose at first that $X = \emptyset$. If there are no arrows from $A$ to $B$, then $L(A, B, \emptyset) = \{\varepsilon\}$ when $A = B$ and $L(A, B, \emptyset) = \emptyset$ when $A \neq B$; the corresponding regular expressions are $\rho(A, B, \emptyset) = \varepsilon$ and $\rho(A, B, \emptyset) = \emptyset$.

If $X = \emptyset$ and if there are arrows from $A$ to be labelled $x_1, x_2, \ldots, x_k$, then $\rho(A, B, \emptyset) = x_1 + x_2 + \cdots + x_k$.

Now suppose that $|X| > 0$. Choose $C \in X$ and put $Y = X \setminus \{C\}$. By induction we can find regular expressions $\rho(A, B, Y)$, $\rho(A, C, Y)$, $\rho(C, C, Y)$ and $\rho(C, B, Y)$ which define the languages $L(A, B, Y)$, $L(A, C, Y)$, $L(C, C, Y)$ and $L(C, B, Y)$, respectively.
We now put

\[ \rho(A, B, X) = \rho(A, B, Y) + \rho(A, C, Y) \rho(C, C, Y)^* \rho(C, B, Y) \]

and claim that this is a regular expression which defines \( L(A, B, X) \). It is a regular expression simply because it is built up from other regular expressions using the operations of addition, concatenation and \( * \)-closure. To see that it defines the language \( L(A, B, X) \), note that for any path that goes from \( A \) to \( B \) via states in \( X \) there are two possibilities: either it goes from \( A \) to \( B \) without using \( C \), in which case its label is in \( L(A, B, Y) \); or else it uses \( C \) at least once, in which case its label is in the language

\[ L(A, C, Y) L(C, C, Y)^* L(C, B, Y). \]

This completes the induction step.

If \( I \) is the initial state, then the regular expression which defines \( L(M) \) is the sum of the regular expressions \( \rho(I, A, S) \), where \( A \) ranges over the accepting states.
The subset construction

We shall now prove that (2) implies (1). That is, given an NFA $M$ we can construct a DFA which accepts the same language.

Let $S$ be the set of states of $M$ and for $X \subseteq S$, let $\overline{X}$ be the set $X$ together with all states which can be reached from a state in $X$ by a path all of whose arrows are labelled with $\varepsilon$. We call $\overline{X}$ the $\varepsilon$-closure of $X$.

From the construction we have $\overline{\overline{X}} = \overline{X}$. We define a DFA $\overline{M}$ as follows: The alphabet of $\overline{M}$ is the alphabet $\Sigma$ of $M$. The states of $\overline{M}$ are the subsets $X \subseteq S$ such that $\overline{X} = X$. For $a \in \Sigma$ and new state $X$ we define a transition $X \xrightarrow{a} Y$, where $Y$ is the $\varepsilon$-closure of the set of states reachable from a state of $X$ by a path labelled $a$. If $I$ is the start state of $M$, then $\overline{I}$ is the start state of $\overline{M}$.

The accepting states of $\overline{M}$ are the new states $X$ which contain an accepting state of $M$.
Rather than finding all $\varepsilon$-closed subsets of $S$ it is enough to begin with $I$ and construct only those new states which are needed for the transitions. For example, if we begin with the NFA

then the subset construction produces the DFA

![Diagram](image-url)