In an earlier chapter we considered some examples of FSA. Consider the example serial adder. After getting some input, the machine can be in ‘carry’ state or ‘no carry’ state. It does not matter what exactly the earlier input was. It is only necessary to know whether it has produced a carry or not. Hence the finite state automaton need not distinguish between each and every input. It distinguishes between classes of inputs.
In the above case, the whole set of inputs can be partitioned into two classes - one that produces a carry and another that does not produce a carry. Thus the finite state automaton distinguishes between classes of input strings. These classes are also finite. Hence we say that the FSA has finite amount of memory.
**Theorem** The following three statements are equivalent.

1. $L \subseteq \Sigma^*$ is accepted by a DFSA.

2. $L$ is the union of some of the equivalence classes of a right invariant equivalence relation of finite index on $\Sigma^*$.

3. Let equivalence relation $R_L$ be defined over $\Sigma^*$ as follows: $xR_Ly$ if and only if, $\forall z \in \Sigma^*$, $xz$ is in $L$ exactly when $yz$ is in $L$. Then $R_L$ is of finite index.
Proof

We shall prove \((1) \Rightarrow (2), (2) \Rightarrow (3), (3) \Rightarrow (1)\).
\((1) \Rightarrow (2)\).

Let \(L\) be accepted by a FSA \(M = (K, \Sigma, \delta, q_0, F)\).

Define a relation \(R_M\) on \(\Sigma^*\) such that \(x R_M y\) if \(\delta(q_0, x) = \delta(q_0, y)\). \(R_M\) is an equivalence relation, as seen below.

\(\forall x \ x R_M x\), since \(\delta(q_0, x) = \delta(q_0, x)\),

\(\forall x \ x R_M y \Rightarrow y R_M x \therefore \delta(q_0, x) = \delta(q_0, y)\) means \(\delta(q_0, y) = \delta(q_0, x)\),

\(\forall x, y \ x R_M y \text{ and } y R_M z \Rightarrow x R_M z. \text{ For if } \delta(q_0, x) = \delta(q_0, y) \text{ and } \delta(q_0, y) = \delta(q_0, z) \text{ then } \delta(q_0, x) = \delta(q_0, z)\).
So \( R_M \) divides \( \Sigma^* \) into equivalence classes. The set of strings which take the machine from \( q_0 \) to a particular state \( q_i \) are in one equivalence class. The number of equivalence classes is therefore equivalent to the number of states of \( M \), assuming every state is reachable from \( q_0 \). (If a state is not reachable from \( q_0 \), it can be removed without affecting the language accepted). It can be easily seen that this equivalence relation \( R_M \) is right invariant, i.e., if 
\[ xR_M y, \quad xzR_M yz \quad \forall z \in \Sigma^*. \]
\[ \delta(q_0, x) = \delta(q_0, y) \text{ if } xR_M y, \]
\[ \delta(q_0, xz) = \delta(\delta(q_0, x), z) = \delta(\delta(q_0, y), z) = \delta(q_0, yz). \]
Therefore \( xzR_M yz \).
$L$ is the union of those equivalence classes of $R_M$ which correspond to final states of $M$.

$(2) \implies (3)$

Assume statement $(2)$ of the theorem and let $E$ be the equivalence relation considered. Let $R_L$ be defined as in the statement of the theorem. We see that $xEy \implies xR_Ly$.

If $xEy$, then $xzEy z$ for each $z \in \Sigma^*$. $xz$ and $yz$ are in the same equivalence class of $E$. Hence $xz$ and $yz$ are both in $L$ or both not in $L$ as $L$ is the union of some of the equivalence classes of $E$. Hence $xR_Ly$.

Hence any equivalence class of $E$ is completely contained in an equivalence class of $R_L$. Therefore $E$ is a refinement of $R_L$ and so the index of $R_L$ is less than or equal to the index of $E$ and hence finite.
(3) ⇒ (1)

First we show $R_L$ is right invariant. $xR_Ly$ if $\forall z$ in $\Sigma^*$, $xz$ is in $L$ exactly when $yz$ is in $L$ or we can also write this in the following way: $xR_Ly$ if for all $w$, $z$ in $\Sigma^*$, $xwz$ is in $L$ exactly when $ywz$ is in $L$. If this holds $xwR_Lyw$.

Therefore $R_L$ is right invariant. Let $[x]$ denote the equivalence class of $R_L$ to which $x$ belongs.

Construct a DFSA $M_L = (K', \Sigma, \delta', q_0, F')$ as follows: $K'$ contains one state corresponding to each equivalence class of $R_L$. $[\epsilon]$ corresponds to $q_0'$. $\delta'$ is defined as follows: $\delta'([x], a) = [xa]$. This definition is consistent as $R_L$. 

Introduction to Formal Languages, Automata and Computability – p.8/23
is right invariant. Suppose $x$ and $y$ belong to the same equivalence class of $R_L$. Then $xa$ and $ya$ will belong to the same equivalence class of $R_L$. For,

$$
\delta'([x], a) = \delta'([y], a)
$$

$$
\downarrow \quad \downarrow
$$

$$
[xa] = [ya]
$$

if $x \in L$, $[x]$ is a final state in $M'$, i.e., $[x] \in F'$. This automaton $M'$ accepts $L$. 

Example

Consider the FSA $M$ given in next figure.

The language accepted consists of strings of $a$’s and $b$’s having at least one $a$. $M$ divides $\{a, b\}^*$ into 3 equivalence classes.

1. $H_1$, set of strings which take $M$ from $q_0$ to $q_0$ i.e., $b^*$.

2. $H_2$, set of strings which take $M$ from $q_0$ to $q_1$, i.e., set of strings which have odd numbers of $a$’s.
3. $H_3$, set of strings which take $M$ from $q_0$ to $q_2$, i.e., set of strings which have even number of $a$’s.

$L = H_2 \cup H_3$ as can be seen.

1. Let $x \in H_1$ and $y \in H_2$. Then $xb \in H_1$ and $yb \in H_2$. Then $xb \notin L$ and $yb \in L$. Therefore $x \not\in_R L y$.

2. Let $x \in H_1$ and $y \in H_3$. Then $xb \in H_1$ and so $xb \notin L$ and $yb \in H_3$ and so $xb \in L$. Therefore $x \not\in_R L y$.

3. Let $x \in H_2$ and $y \in H_3$. Take any string $z$, $xz$ belongs to either $H_2$ or $H_3$ and so in $L$, $yz$ belongs to either $H_2$ or $H_3$ and so in $L$. Therefore $x \not\in_R L y$. 

Introduction to Formal Languages, Automata and Computability – p.11/23
So if we construct $M'$ as in the proof of the theorem, we have one state corresponding to $H_1$ and one state corresponding to $L = H_2 \cup H_3$.

This is the automaton we get as $M'$. We see that, it accepts $L(M)$. Both $M$ and $M'$ are DFSA accepting the same language. But $M'$ has minimum number of states and is called the minimum state automaton.
Theorem The minimum state automaton accepting a regular set $L$ is unique up to an isomorphism and is given by $M'$ in the proof of previous theorem.

In the proof of previous theorem, we started with $M$, found equivalence classes for $R_M$, $R_L$ and constructed $M'$. The number of states of $M$ is equal to the index of $R_M$ and the number of states of $M'$ is equal to the index of $R_L$. Since $R_M$ is a refinement of $R_L$, the number of states of $M'$ is less than or equal to the number of states of $M$. If $M$ and $M'$ have the same number of states, then we can find a mapping $h : K \rightarrow K'$.
(which identifies each state of $K$ with a state of $K'$) such that if $h(q) = q'$ then for $a \in \Sigma$, 

$$h(\delta(q, a)) = \delta'(q', a).$$

This is achieved by defining $h$ as follows: $h(q_0) = q'_0$ and if $q \in K$, then there exists a string $x$ such that $\delta(q_0, x) = q$. $h(q) = q'$ where $\delta(q'_0, x) = q'$. This definition of $h$ is consistent. This can be seen as follows: Let $\delta(q, a) = p$ and $\delta'(q', a) = p'$, $\delta(q_0, xa) = p$ and $\delta'(q'_0, xa) = p'$ and hence $h(p) = p'$. 
Minimization of DFSA

Let $M = (K, \Sigma, \delta, q_0, F)$ be a DFSA. Let $R$ be an equivalence relation on $K$ such that $pRq$, if and only if for each input string $x$, $\delta(p, x) \in F$ if and only if $\delta(q, x) \in F$. This essentially means that if $p$ and $q$ are equivalent, then either $\delta(p, x)$ and $\delta(q, x)$ both are in $F$ or both are not in $F$ for any string $x$. $p$ is distinguishable from $q$ if there exists a string $x$ such that one of $\delta(q, x)$, $\delta(p, x)$ is in $F$ and the other is not. $x$ is called the distinguishing string for the pair $< p, q >$.

If $p$ and $q$ are equivalent $\delta(p, a)$ and $\delta(q, a)$ will be equivalent for any $a$. If $\delta(p, a) = r$ and $\delta(q, a) = s$ and $r$ and $s$ are distinguishable by $x$, then $p$ and $q$ are distinguishable by $ax$. 

Introduction to Formal Languages, Automata and Computability – p.15/23
Algorithm to find minimum DFSA

We get a partition of the set of states of $K$ as follows:

**Step 1** Consider the set of states in $K$. Divide them into two blocks $F$ and $K - F$. (Any state in $F$ is distinguishable from a state in $K - F$ by $\epsilon$)

Repeat the following step till no more split is possible.

**Step 2** Consider the set of states in a block. Consider the $a$-successors of them for $a \in \Sigma$. If they belong to different blocks, split this block into two or more blocks depending on the $a$-successors of the states.

For example if a block has $\{q_1, \ldots, q_k\}$. $\delta(q_1, a) = p_1$, $\delta(q_2, a) = p_2$, $\ldots$, $\delta(q_k, a) = p_k$ and $p_1, \ldots, p_i$ belong to one block, $p_{i+1}, \ldots, p_j$ belong to another block and
$p_{j+1}, \ldots, p_k$ belong to third block, then split
$\{q_1, \ldots, q_k\}$ into $\{q_1, \ldots, q_i\} \{q_{i+1}, \ldots, q_j\}$
$\{q_{j+1}, \ldots, q_k\}$.

**Step 3** For each block $B_i$, consider a state $b_i$.
Construct $M' = (K', \Sigma, \delta', q'_0, F')$ where $K' = \{b_i \mid B_i$
$\text{is a block of the partition obtained in step 2}\}$. $q'_0$ corresponds to the block containing $q_0$.
$\delta(b_i, a) = b_j$ if there exits $q_i \in B_i$ and $q_j \in B_j$ such
that $\delta(q_i, a) = q_j$. $F'$ consists of states corresponding
to the blocks containing states in $F$. 
Consider the following FSA $M$ over $\Sigma = \{b, c\}$ accepting strings which have $bcc$ as a substrings. A nondeterministic automaton for this will be,

Converting to DFSA we get $M'$ as in next figure.
where, $p_0 = [q_0]$  $p_1 = [q_0, q_1]$  $p_2 = [q_0, q_2]$
$p_3 = [q_0, q_3]$  $p_4 = [q_0, q_1, q_3]$
$p_5 = [q_0, q_2, q_3]$. Finding the minimum state automaton for $M'$

**Step 1**  Divide the set of states into 2 blocks

$\overline{p_0p_1p_2}$  $\overline{p_3p_4p_5}$.

In $\overline{p_0p_1p_2}$, the $b$ successors are in one block, the $c$ successors of $p_0p_1$ are in one block and $p_2$ is in another block. Therefore $\overline{p_0p_1p_2}$ is split as $\overline{p_0p_1}$ and $\overline{p_2}$. The $b$ and $c$ successors of $\overline{p_3p_4p_5}$ are in the same block.
Now the partition is $p_0p_1 \ p_2 \ p_3p_4p_5$. Consider $p_0p_1$. The $b$ successors of $p_0p_1$ are in the same block but the $c$ successors of $p_0$ and $p_1$ are $p_0$ and $p_2$ and they are in different blocks. Therefore $p_0p_1$ is split into $p_0$ and $p_1$. Now the partition is $p_0 \ p_1 \ p_2 \ p_3p_4p_5$. No further split is possible. The minimum state automaton is
The minimization procedure cannot be applied to NFSA. For example consider the NFSA

The language accepted is represented by the regular expression \((0 + 1)^*0\) for which NFSA in next figure will
suffice. But if we try to use the minimization procedure $q_0q_1q_2$ will be initially split as $q_0q_2$ and $q_1$. $q_0$ and $q_2$ are not equivalent as $\delta(q_0, 0)$ contains a final state while $\delta(q_2, 0)$ does not. So they have to be split and the FSA in first figure cannot be minimized using the minimization procedure.

Myhill-Nerode theorem can also be used to show that certain sets are not regular.
Example

We know $L = \{a^n b^n | n \geq 1\}$ is not regular. Suppose $L$ is regular. Then by Myhill-Nerode theorem, $L$ is the union of the some of the equivalence classes of a right invariant relation $\equiv$ over $\{a, b\}$. Consider $a, a^2, a^3, \ldots$ since the number of equivalence classes is finite, for some $m$ and $n$, $m \neq n$, $a^m$ and $a^n$ must be in the same equivalence class. We write this as $a^m \equiv a^n$. Since $\equiv$ is right invariant $a^m b^m \equiv a^n b^m$. i.e., $a^m b^m$ and $a^n b^m$ are in the same equivalence class. $L$ either contains one equivalence class completely or does not contain that class. Hence since $a^m b^m \in L$, $L$ should contain this class completely and hence $a^n b^m \in L$ which is a contradiction. Therefore $L$ is not regular.