

Once we have a flow table or state table for our FSM we may need to do some simplification in order to get an efficient circuit.

In general, we would like to produce an optimal flow table which implements the same sequential function as the given flow table but with as few states as possible.

Definition: Two flow tables are said to be **equivalent** or **indistinguishable** if they produce identical output sequences for the same input sequence.

We first assume that we have a completely specified flow table defining the sequential circuit's output behavior, i.e., don't cares do not appear in the table. This is often the case when we are designing a finite state machine which is to operate in pulse mode with either level or pulse outputs.

There are two types of states we are interested in removing from the flow table:

- unreachable (inaccessible) states
- equivalent (indistinguishable) states

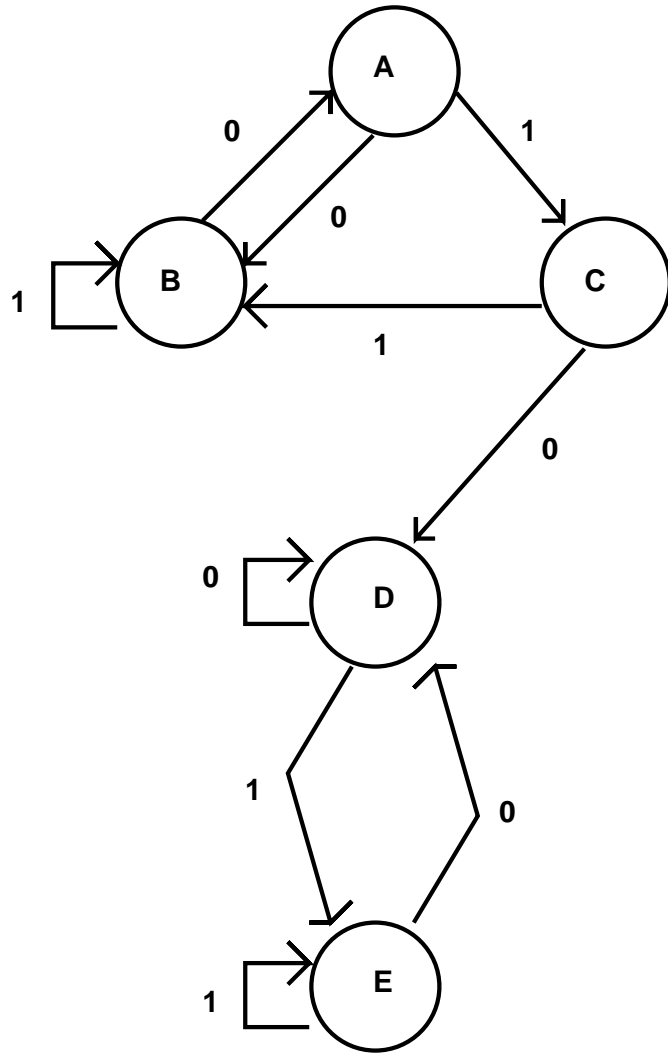
Unreachable States

Definition: State S_1 is **reachable** from state S_2 in a machine M if there exists an input sequence which takes M from S_2 to S_1 .

Definition: A state S in a machine M is **unreachable** or **inaccessible** if and only if either

1. there is no input sequence that takes the machine from its initial state to state S , or
2. if there is no initial state specified, there is at least one other state T for which there is no input sequence that causes the machine to reach state S .

The reachable and unreachable states are easiest to see by considering the state diagrams. We will assume that all unreachable states have been removed from the machine and therefore from the flow table.



Initial state = D or E then A,B,C are unreachable

Initial state = A or B or C then all are reachable

No initial state then A,B,C are unreachable

Equivalent or Indistinguishable States

Definition: States S_i and S_j are said to be **distinguishable** if and only if there exists at least one finite input sequence for which the machine M produces a different output sequence depending upon which state, S_i or S_j is used as the initial state.

Definition: A sequence that shows S_i and S_j are distinguishable is called a distinguishing sequence of (S_i, S_j) .

Definition: If there exists a distinguishing sequence of length k for states S_i and S_j the states are called k -distinguishable.

state	$\langle ck \rangle \uparrow$	
	$x = 0$	$x = 1$
A	E,0	D,1
B	F,0	D,0
C	E,0	B,1
D	F,0	B,0
E	C,0	F,1
F	B,0	C,0

For the Mealy machine above

- (A, B) are 1-distinguishable via $x = 1$
 - $A \xrightarrow{1} D$
 - $B \xrightarrow{0} D$
- (A, E) are not 1-distinguishable or 2-distinguishable but are 3-distinguishable via $x = 1, 1, 1$
 - $A \xrightarrow{1} D \xrightarrow{0} B \xrightarrow{0} D$
 - $E \xrightarrow{1} F \xrightarrow{0} C \xrightarrow{1} B$

Definition: If there does not exist a distinguishing sequence of length k for two states S_i and S_j then they are called k -equivalent.

Definition: S_i and S_j are said to be **equivalent** or **indistinguishable** if and only if for every possible input sequence, the same output sequence is produced regardless of whether S_i or S_j is the initial state. Equivalence is denoted $S_i \equiv S_j$.

So what?

If the initial states of two machines are equivalent then the machines are equivalent.

In a given machine, if S_i and S_j are equivalent then one can be used to replace the other and produce a new flow table which is indistinguishable from the old but with one less state and therefore closer to optimal.

Suppose we could partition the states into disjoint sets of states, C_1, \dots, C_p , such that any two states in the same set were indistinguishable and any two states in different sets were distinguishable.

If $S_i \in C_r$ and $S_j \in C_t$ then

$$\begin{aligned} r = t &\rightarrow S_i \equiv S_j \\ r \neq t &\rightarrow S_i \not\equiv S_j \end{aligned}$$

Therefore, by choosing one state from each set $S_{j_r} \in C_r$ and replacing all other states that are in C_r with S_{j_r} in the flow table we produce a new flow table with just p states.

Our goal is to find a way to produce C_1, \dots, C_p with minimum p .

We will discuss two ways to do this

- the pair chart method
- the partition method (Moore reduction procedure)

Pair Method

The pair method is based on the following facts:
state indistinguishability is an **equivalence relation**

- $A \equiv A$ (symmetric)
- $A \equiv B \rightarrow B \equiv A$ (reflexive)
- $A \equiv B, B \equiv C \rightarrow A \equiv C$

Theorem: States A and B are indistinguishable if and only if

- they have identical outputs and
- their 0-successors are indistinguishable
- their 1-successors are indistinguishable

Of course, for multibit inputs these conditions become all corresponding next state entries in the table are indistinguishable. For example, for a two-bit input symbol we have their 00-successors are indistinguishable; their 01-successors are indistinguishable; their 10-successors are indistinguishable; and their 11-successors are indistinguishable.

Step 0: Form an array whose cells correspond to pairs of states and mark as distinguishable any pairs that have different outputs. Also mark the upper half of the table since it is redundant due to reflexivity (marked in the table below with an R).

state	$x = 0$	$x = 1$
A	E,0	D,1
B	F,0	D,0
C	E,0	B,1
D	F,0	B,0
E	C,0	F,1
F	B,0	C,0

B	X	R	R	R	R
C		X	R	R	R
D	X		X	R	R
E		X		X	R
F	X		X		X
	A	B	C	D	E

Now add in each cell the 0-successors for the pair and the 1-successors for the pair.

B	X	R	R	R	R
C	EE,BD	X	R	R	R
D	X	FF,BD	X	R	R
E	CE,FD	X	CE,FB	X	R
F	X	BF,CD	X	BF,CB	X
	A	B	C	D	E

If any pairs have identical next states they can be marked as equivalent (there are none in this table).

Now mark as distinguishable any cell that points to a cell already marked as distinguishable.

B	X	R	R	R	R
C	EE,BD	X	R	R	R
D	X	FF,BD	X	R	R
E	CE,FD	X	CE,FB	X	R
F	X	X	X	X	X
	A	B	C	D	E

We have marked BF due to cell CD and marked DF due to cell CB.

Repeat the process since new distinguishable pairs were found.

B	X	R	R	R	R
C	EE,BD	X	R	R	R
D	X	FF,BD	X	R	R
E	X	X	X	X	R
F	X	X	X	X	X
	A	B	C	D	E

AE marked due to cell FD and CE marked due to cell BF.

No more new marks are possible so we have the following

$\{AC\}$, $\{BD\}$, $\{E\}$ and $\{F\}$.

In general, you may have to apply transitivity to group equivalent states together after the final array is produced.

For example, suppose you have the following pairs and singletons resulting from applying the chart method to a FSM with 8 states originally.

(1) (2) (3, 4) (5, 6) (5, 7) (5, 8) (6, 7) (6, 8) (7, 8)

After applying transitivity this becomes

(1) (2) (3, 4) (5, 6, 7, 8)

Moore Reduction Procedure

The presentation of this method will follow that of Kohavi. The resulting method is identical to that presented in the text. Unless otherwise stated we will assume that the finite state machine is a Mealy machine.

We will also assume, without loss of generality, that there is a single input variable, x , that can be 0 or 1 at the clock pulse.

Adapting the technique to multibit inputs is straightforward.

The Moore Reduction Procedure is based on an inductive definition of state equivalence based on input sequence length.

As with the pair chart the conditions on successors are extended to all 4 successors for two-bit input symbols, 8 successors for three-bit input symbols etc.

Theorem: States A and B are $k+1$ -equivalent if and only if

- A and B are k -equivalent
- the 0-successor of A and the 0-successor of B are k -equivalent
- the 1-successor of A and the 1-successor of B are k -equivalent

Proof:

(\rightarrow) Assume that states A and B are $k+1$ -equivalent. We must deduce the three k -equivalent statements above.

It follows trivially that A and B must also be k -equivalent.

Let the 0-successors of A and B be R_0 and S_0 respectively and suppose that R_0 and S_0 were k -distinguishable with distinguishing sequence X of length k . The sequence $0, X$ is therefore a distinguishing sequence of A and B of length $k+1$ and therefore A and B are $k+1$ -distinguishable. This is a contradiction. So the 0-successors must be k -equivalent.

The same argument shows that the 1-successors R_1 and S_1 are also k -equivalent.

(\leftarrow)

Now assume that (A, B) , (R_0, S_0) and (R_1, S_1) are all k -equivalent pairs of states.

We must deduce that (A, B) are $k + 1$ -equivalent

(A, B) must therefore be 1-equivalent so any single bit input causes the same output when starting in A or B .

Any $k + 1$ bit input sequence can be written $0, Y$ or $1, Y$ where Y is a k bit input sequence. Consider $0, Y$. The 0 causes no problems since the output is the same for both A and B as the initial state. The new state is R_0 and S_0 respectively. Since (R_0, S_0) is a k -equivalent pair of states the machine behavior when applying Y as next k bits of input is the same regardless of starting at R_0 or S_0 . So A and B produce identical outputs given $0, Y$.

Similarly, they produce identical outputs for $1, Y$ and since this covers all possible input sequences of length $k + 1$ it follows that (A, B) is a pair of $k + 1$ -equivalent states. \square

Let P_1 be a partition of the states of the Mealy machine for which we are given a flow table into sets $C_1^{(1)}, \dots, C_{n_1}^{(1)}$ where if $A \in C_i^{(1)}$ and $B \in C_j^{(1)}$ then

$$\begin{aligned}
 i = j &\rightarrow (A, B) \text{ 1 - equivalent} \\
 i \neq j &\rightarrow (A, B) \text{ 1 - distinguishable}
 \end{aligned}$$

For our example

state	$x = 0$	$x = 1$
A	E,0	D,1
B	F,0	D,0
C	E,0	B,1
D	F,0	B,0
E	C,0	F,1
F	B,0	C,0

$C_1^{(1)} = \{A, C, E\}$ and $C_2^{(1)} = \{B, D, F\}$. Note that this merely distinguishes the states based on their output.

P_2 can be defined in an analogous manner to P_1 but using 2-equivalence.

Can we create P_2 from P_1 via some simple procedure?

We can use the Theorem to create the sets that define P_2 , i.e. $C_1^{(2)}, \dots, C_{n_2}^{(2)}$.

If $A \in C_i^{(2)}$ and $B \in C_j^{(2)}$ then

$i = j \rightarrow (A, B) \text{ 2-equivalent}$

$i \neq j \rightarrow (A, B) \text{ 2-distinguishable}$

Suppose $A \in C_i^{(1)}$ and $B \in C_j^{(1)}$

- $i \neq j \rightarrow (A, B)$ are 2-distinguishable – so no additional set need be created in P_2 .
- $i = j \rightarrow (A, B)$ 2-equivalent or 2-distinguishable depending on their 0-successors and 1-successors – this pair may or may not cause an additional set in P_2 .

If $i = j$ and the 0-successors of (A, B) are 1-equivalent and the 1-successors of (A, B) are 1-equivalent then (A, B) is a 2-equivalent pair of states and $A \in C_m^{(2)}$ and $B \in C_m^{(2)}$.

If $i = j$ and either the 0-successors of (A, B) are not 1-equivalent or the 1-successors of (A, B) are not 1-equivalent then (A, B) is a 2-distinguishable pair of states and $A \in C_r^{(2)}$ and $B \in C_s^{(2)}$ with $r \neq s$. So we must add a set to P_2 and place A and B in either.

So we see that for any $1 \leq j \leq n_2$ there must be an $1 \leq i \leq n_1$ such that $C_j^{(2)} \subseteq C_i^{(1)}$. **we are refining the partition P_1 to get P_2 .**

This gives us a straightforward algorithm for producing P_2 from P_1 .

initialize $j = 0$ and do the following for $i = 1, \dots, n_1$

- evaluate the 0-successors of all states in $C_i^{(1)}$
- evaluate the 1-successors of all states in $C_i^{(1)}$
- group the states of $C_i^{(1)}$ into new sets where all 0-successors associated with the states in a particular new set are 1-equivalent and all 1-successors associated with the states in a particular new set are 1-equivalent. (Suppose there are m_i such new sets from $C_i^{(1)}$.)
- for $r = 1, \dots, m_i$ do the following
 - $j = j + 1$
 - create $C_j^{(2)}$ and place in it the states associated with the r -th new set just created.

state	$x = 0$	$x = 1$
A	E,0	D,1
B	F,0	D,0
C	E,0	B,1
D	F,0	B,0
E	C,0	F,1
F	B,0	C,0

P_1 is given by $C_1^{(1)} = \{A, C, E\}$ and $C_2^{(1)} = \{B, D, F\}$.

- $C_1^{(1)} = \{A, C, E\}$, 1-successors $\{D, B, F\} \rightarrow \{C_2^{(1)}, C_2^{(1)}, C_2^{(1)}\}$,
0-successors $\{E, E, C\} \rightarrow \{C_1^{(1)}, C_1^{(1)}, C_1^{(1)}\}$. So $C_1^{(1)} = \{A, C, E\} = C_1^{(2)}$.
- $C_2^{(1)} = \{B, D, F\}$, 1-successors $\{D, B, C\} \rightarrow \{C_2^{(1)}, C_2^{(1)}, C_1^{(1)}\}$,
0-successors $\{F, F, B\} \rightarrow \{C_2^{(1)}, C_2^{(1)}, C_2^{(1)}\}$. So $C_2^{(2)} = \{B, D\}$
and $C_3^{(2)} = \{F\}$.

state	$x = 0$	$x = 1$
A	E,0	D,1
B	F,0	D,0
C	E,0	B,1
D	F,0	B,0
E	C,0	F,1
F	B,0	C,0

P_3 can be created from P_2 via the same procedure $C_1^{(2)} = \{A, C, E\}$, $C_2^{(2)} = \{B, D\}$ and $C_3^{(2)} = \{F\}$.

- $C_1^{(2)} = \{A, C, E\}$, 0-successors $\{E, E, C\} \rightarrow \{C_1^{(2)}, C_1^{(2)}, C_1^{(2)}\}$, 1-successors $\{D, B, F\} \rightarrow \{C_2^{(2)}, C_2^{(2)}, C_3^{(2)}\}$. So $C_1^{(3)} = \{A, C\}$ and $C_2^{(3)} = \{E\}$.
- $C_2^{(2)} = \{B, D\}$ 0-successors $\{F, F\} \rightarrow \{C_3^{(2)}, C_3^{(2)}\}$, 1-successors $\{D, B\} \rightarrow \{C_2^{(2)}, C_2^{(2)}\}$. So $C_3^{(3)} = \{B, D\}$.
- $C_4^{(3)} = \{F\}$.

Repeat to get P_4

$$C_1^{(3)} = \{A, C\}, C_2^{(3)} = \{E\}, C_3^{(3)} = \{B, D\}, \text{ and } C_4^{(3)} = \{F\}.$$

- $C_1^{(3)} = \{A, C\}$, 0-successors $\{E, E\} \rightarrow \{C_2^{(3)}, C_2^{(3)}\}$ 1-successors $\{D, B\} \rightarrow \{C_3^{(3)}, C_3^{(3)}\}$. So $C_1^{(4)} = \{A, C\}$.
- $C_2^{(4)} = \{E\}$,
- $C_3^{(3)} = \{B, D\}$, 0-successors $\{F, F\} \rightarrow \{C_4^{(3)}, C_4^{(3)}\}$ 1-successors $\{B, D\} \rightarrow \{C_3^{(3)}, C_3^{(3)}\}$. So $C_3^{(4)} = \{B, D\}$.
- $C_4^{(4)} = \{F\}$.

$P_4 = P_3$ so we have achieved convergence, i.e., $P_r = P_3$ for $r \geq 3$.

In general, if $P_k = P_{k+1}$ then P_k is called the **equivalence** partition. The sets $C_1^{(k)}, \dots, C_{n_k}^{(k)}$ separate the states of the machine into equivalence classes:

if $A \in C_i^{(k)}$ and $B \in C_j^{(k)}$ then

$$i = j \rightarrow A \equiv B$$

$$i \neq j \rightarrow A \not\equiv B$$

Theorem: The equivalence partition P_k comprising the sets $C_1^{(k)}, \dots, C_{n_k}^{(k)}$ is unique.

Therefore this partition can be used to generate the minimal realization in terms of the number of states of the sequential machine specified by the original flow table.

Theorem: If A and B are distinguishable states for a Mealy machine with n states, then they are distinguishable by an input sequence of length $j \leq n - 1$.

The entire procedure can also be applied to a Moore machine. Note however that since the output is associated with a state P_0 rather than P_1 separates the initial states based on output. It also follows that the bound on the distinguishing sequence of two states drops to $n - 2$ in the latter theorem above.

state	$x = 0$	$x = 1$	Z
A	B	C	1
B	D	E	1
C	F	D	0
D	G	C	0
E	H	F	0
F	G	I	0
G	B	J	1
H	J	I	1
I	A	I	0
J	I	D	0

$$P_0 = \{A, B, G, H\} \text{ and } \{C, D, E, F, I, J\}$$

$$P_1 = \{A, G\}, \{B, H\}, \{C, J\}, \{D, E, F, I\}$$

$$P_2 = \{A, G\}, \{B\}, \{H\}, \{C, J\}, \{D\}, \{E\}, \{F, I\}$$

$$P_3 = \{A, G\}, \{B\}, \{H\}, \{C, J\}, \{D\}, \{E\}, \{F, I\}$$

The new machine with 7 states rather than 10 is given by

state	$x = 0$	$x = 1$	Z
(AG) a	b	c	1
(B) b	d	e	1
(CJ) c	f	d	0
(D) d	a	c	0
(E) e	h	f	0
(FI) f	a	f	0
(H) h	c	f	1