Reducing the number of states (state equivalence):

Row reduction method

If you are "careless" in creating your state graph/table, you may end up with states that are equivalent to one another. For equivalent states, it doesn't matter which state you in, you will have the same resulting output for any possible input.

Reducing the number of states might mean fewer flip-flops, but even if not you will have more don't-cares in the state table resulting in less/simpler logic

Example: Previous problem of detecting sequences of possibly overlapping sequences of 0010 or 0001 -- Mealy Design

meaning	PS	NS X=0	NS X=1	Z X=0	Z X=1
RESET	A	B	C	0	0
0	B	D	E	0	0
1	C	F	G	0	0
00	D	H	I	0	0
01	E	~~J~~ F	~~K~~ G	0	0
10	F	~~L~~ D	~~M~~ E	0	0
11	G	~~N~~ F	~~O~~ G	0	0
000	H	H	I	0	1
001	I	~~J~~ F	~~K~~ K	1	0
010	~~J~~	~~L~~ D	~~M~~ E	~~0~~	~~0~~
011	~~K~~	~~N~~ F	~~O~~	~~0~~	~~0~~
100	~~L~~	~~H~~	~~I~~	~~0~~	~~0~~
101	~~M~~	~~J~~	~~K~~	~~0~~	~~0~~
110	~~N~~	~~L~~ D	~~M~~ E	~~0~~	~~0~~
111	~~O~~	~~N~~ F	~~O~~	~~0~~	~~0~~

Row reduction:

Two states are equivalent if they have the same N.S. and same output for any input

D ≡ L

E ≡ M

F ≡ J ≡ N

G ≡ K ≡ O

Minimizing the number of states – row reduction

Here is a methodical way to set up a state table. We'll use the previous example of a Mealy machine sequence detector that will detect possibly overlapping sequences of 0010 or 0001. We will write a state table that "remembers" all possible input sequences of up to the last three values.

Meaning	*PS*	*NS*		*Z*	
		X=0	*X=1*	*X=0*	*X=1*
Reset	*A*	*B*	*C*	*0*	*0*
0	*B*	*D*	*E*	*0*	*0*
1	*C*	*F*	*G*	*0*	*0*
0 0	*D*	*H*	*I*	*0*	*0*
0 1	*E*	*J*	*K*	*0*	*0*
1 0	*F*	*L*	*M*	*0*	*0*
1 1	*G*	*N*	*O*	*0*	*0*
0 0 0	*H*	*H*	*I*	*0*	*1*
0 0 1	*I*	*J*	*K*	*1*	*0*
0 1 0	*J*	*L*	*M*	*0*	*0*
0 1 1	*K*	*N*	*O*	*0*	*0*
1 0 0	*L*	*H*	*I*	*0*	*0*
1 0 1	*M*	*J*	*K*	*0*	*0*
1 1 0	*N*	*L*	*M*	*0*	*0*
1 1 1	*O*	*N*	*O*	*0*	*0*

Note that the "Next State" part of the table is very systematic and it is the same for any state machine that needs to "remember" the most recent three values of a single input.

But this table contains far more states than we need. A first pass technique to reduce the number of states is to note that two states are equivalent if they have the same next state and the same output for every input. By this definition, the following states are equivalent in this table

D ≡ L (not equivalent to state H as output is different)
E ≡ M (not equivalent to state I as output is different)
F ≡ J ≡ N
G ≡ K ≡ O

Equivalent states can be combined by eliminating all but one of a set and then replacing references to the eliminated states with the equivalent state that remains.

So we will :

eliminate state L and replace every instance of it in the table with D
eliminate state M and replace every instance of it in the table with E
eliminate states J and N and replace every instance of them in the table with F
eliminate states K and O and replace every instance of them in the table with G

Meaning	***PS***	***NS***		***Z***	
		X=0	***X=1***	***X=0***	***X=1***
Reset	*A*	*B*	*C*	*0*	*0*
0	*B*	*D*	*E*	*0*	*0*
1	*C*	*F*	*G*	*0*	*0*
0 0	*D*	*H*	*I*	*0*	*0*
0 1	*E*	~~*J*~~ *F*	~~*K*~~ *G*	*0*	*0*
1 0	*F*	~~*L*~~ *D*	~~*M*~~ *E*	*0*	*0*
1 1	*G*	~~*N*~~ *F*	~~*O*~~ *G*	*0*	*0*
0 0 0	*H*	*H*	*I*	*0*	*1*
0 0 1	*I*	~~*J*~~ *F*	~~*K*~~ *G*	*1*	*0*

Redrawing to make it neater:

Meaning	***PS***	***NS***		***Z***	
		X=0	***X=1***	***X=0***	***X=1***
Reset	*A*	*B*	*C*	*0*	*0*
0	*B*	*D*	~~*E*~~ C	*0*	*0*
1	*C*	~~*F*~~ B	~~*G*~~ C	*0*	*0*
0 0	*D*	*H*	*I*	*0*	*0*
~~*0 1*~~	~~*E*~~	~~*F*~~ B	~~*G*~~	~~*0*~~	~~*0*~~
~~*1 0*~~	~~*F*~~	~~*D*~~	~~*E*~~	~~*0*~~	~~*0*~~
~~*1 1*~~	~~*G*~~	~~*F*~~ B	~~*G*~~	~~*0*~~	~~*0*~~
0 0 0	*H*	*H*	*I*	*0*	*1*
0 0 1	*I*	~~*F*~~ B	~~*G*~~ C	*1*	*0*

With the new renamed states, we see the following are now equivalent as they go to the same next state and have the same output for every input.

B ≡ F

C ≡ E ≡ G (not equivalent to state I as output is different)

So we will :

eliminate state F and replace every instance of it in the table with B
eliminate states E and G and replace every instance of them in the table with C

Meaning	*PS*	*NS*		*Z*	
		X=0	*X=1*	*X=0*	*X=1*
Reset	*A*	*B*	*C*	*0*	*0*
0	*B*	*D*	~~*E*~~ *C*	*0*	*0*
1	*C*	~~*F*~~ *B*	~~*G*~~ *C*	*0*	*0*
0 0	*D*	*H*	*I*	*0*	*0*
0 0 0	*H*	*H*	*I*	*0*	*1*
0 0 1	*I*	~~*F*~~ *B*	~~*G*~~ *C*	*1*	*0*

Redrawing to make it neater:

Meaning	*PS*	*NS*		*Z*	
		X=0	*X=1*	*X=0*	*X=1*
Reset	*A*	*B*	*C*	*0*	*0*
0	*B*	*D*	~~*C*~~ A	*0*	*0*
1	~~*C*~~	~~*B*~~	~~*C*~~	~~*0*~~	~~*0*~~
0 0	*D*	*H*	*I*	*0*	*0*
0 0 0	*H*	*H*	*I*	*0*	*1*
0 0 1	*I*	*B*	~~*C*~~ A	*1*	*0*

With the new renamed states, we see the following are now equivalent as they go to the same next state and have the same output for every input.

A ≡ C (not equivalent to state I as the output is different)

So we will :

eliminate state C and replace every instance of it in the table with A

Meaning	*PS*	*NS*		*Z*	
		X=0	*X=1*	*X=0*	*X=1*
Reset	*A*	*B*	~~*C*~~ *A*	*0*	*0*
0	*B*	*D*	~~*C*~~ *A*	*0*	*0*
0 0	*D*	*H*	*I*	*0*	*0*
0 0 0	*H*	*H*	*I*	*0*	*1*
0 0 1	*I*	*B*	~~*C*~~ *A*	*1*	*0*

Redrawing to make it neater:

Meaning	***PS***	***NS***		***Z***	
		X=0	***X=1***	***X=0***	***X=1***
Reset	*A*	*B*	*A*	*0*	*0*
0	*B*	*D*	*A*	*0*	*0*
0 0	*D*	*H*	*I*	*0*	*0*
0 0 0	*H*	*H*	*I*	*0*	*1*
0 0 1	*I*	*B*	*A*	*1*	*0*

At this point there is no further minimization we can do using the row reduction method and this becomes our final state table.

Note that in some cases we can reduce further by using an implication table or other similar method. These methods make use of the fact that two states are equivalent if they go to *equivalent* next states (not necessarily the *same* next state) and have the same output for every input.