FSM Decomposition

Often convenient to realize a sequential circuit as an inter-connection of subcircuits that realize the same terminal behavior.

First, decide on how the overall circuit is to be broken up and what function each of the sub-circuits must serve

Then, treat each of the sub-circuits as a separate and independent design problem.

- Such an approach is more or less required for large problems.

Uses of Decomposition

Breaks up original problem into smaller ones.

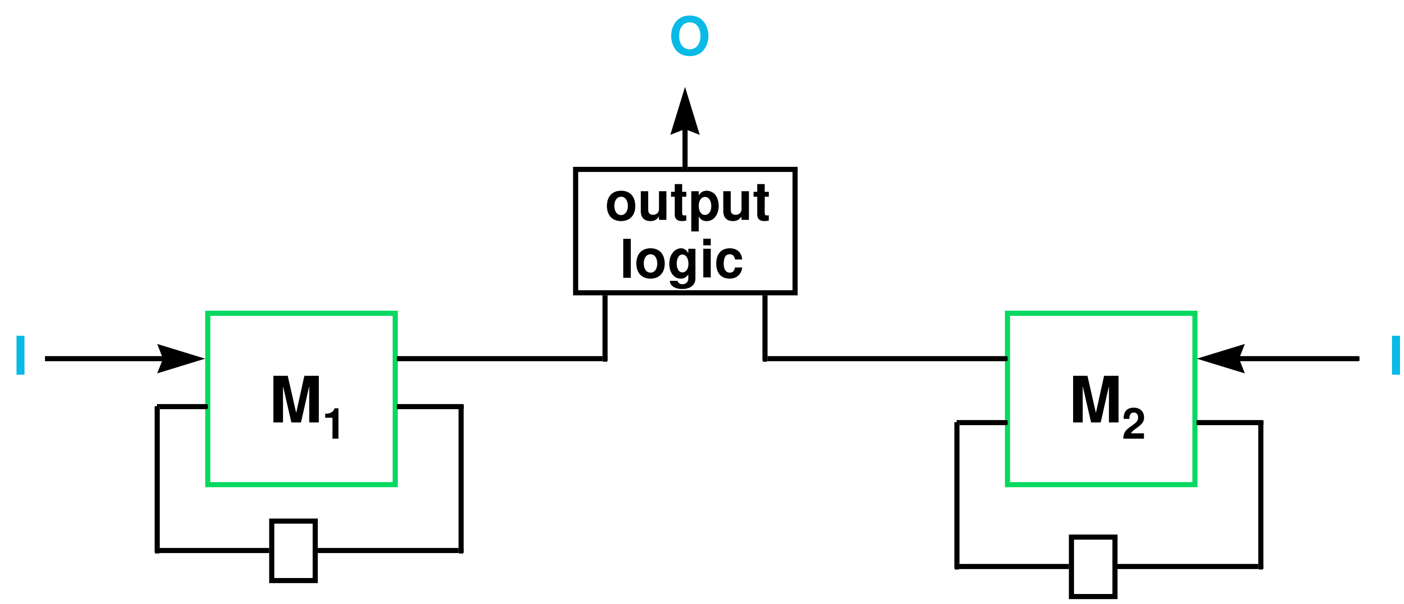
Heuristics for state assignment and logic optimization work better on smaller problems as opposed to larger ones

$\Rightarrow$ decomposition may lead to more economical realizations in terms of area.

Performance of decomposed design may be better due to smaller critical path delay.

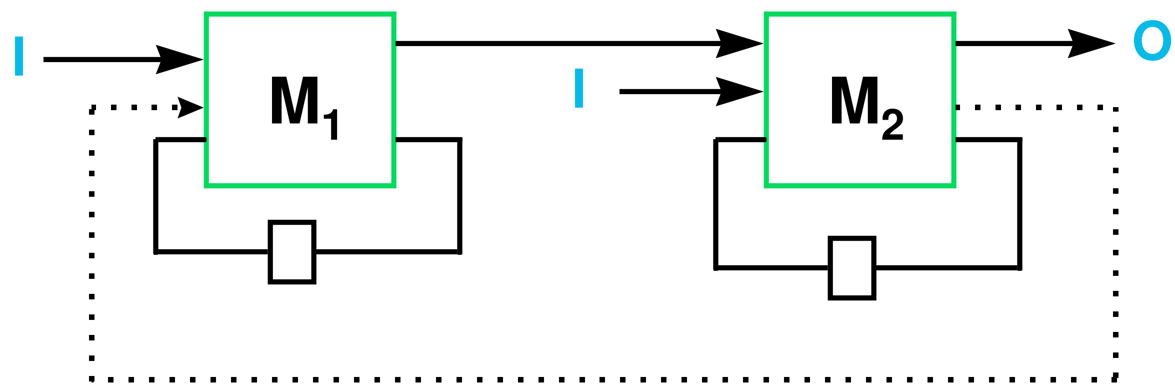
Forms of Decomposition

Parallel decomposition: Both machines operate independently.



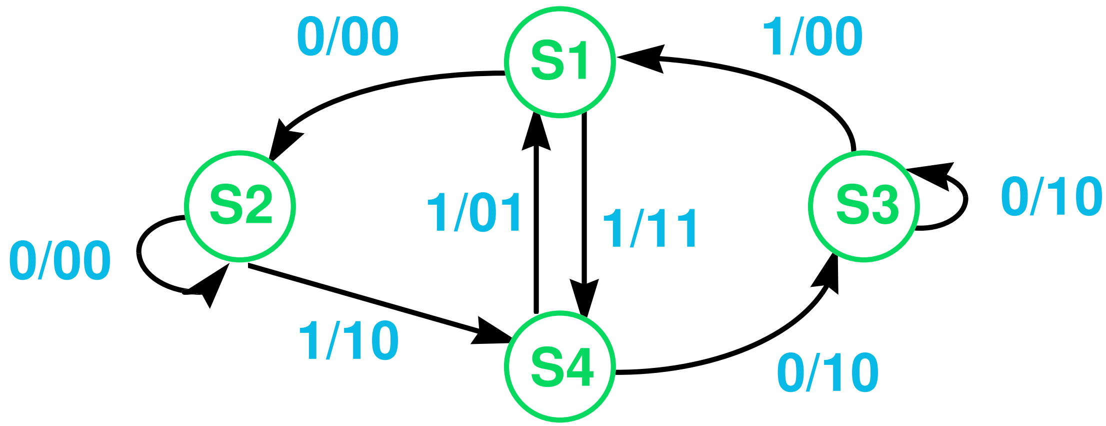
Forms of Decomposition - II

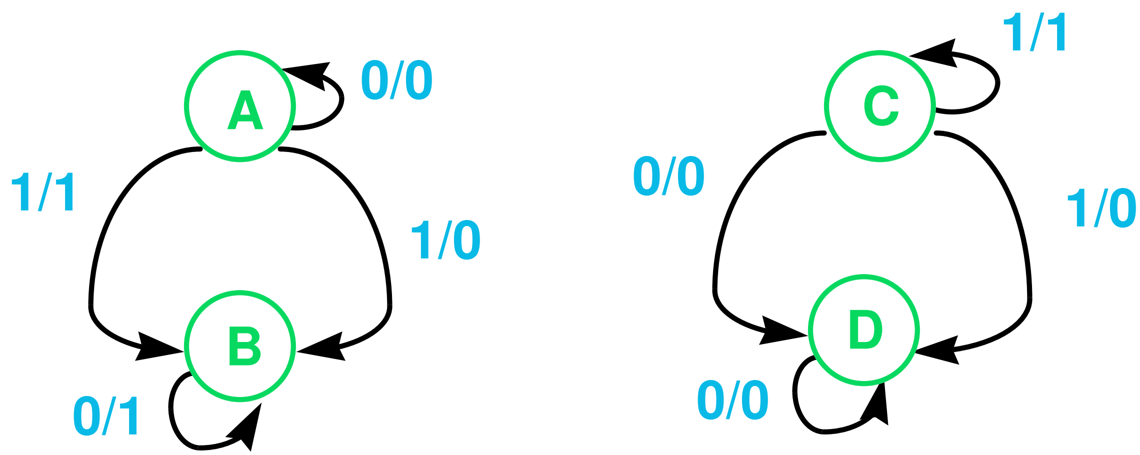
Cascade decomposition: Tail machine "knows" state of head machine.



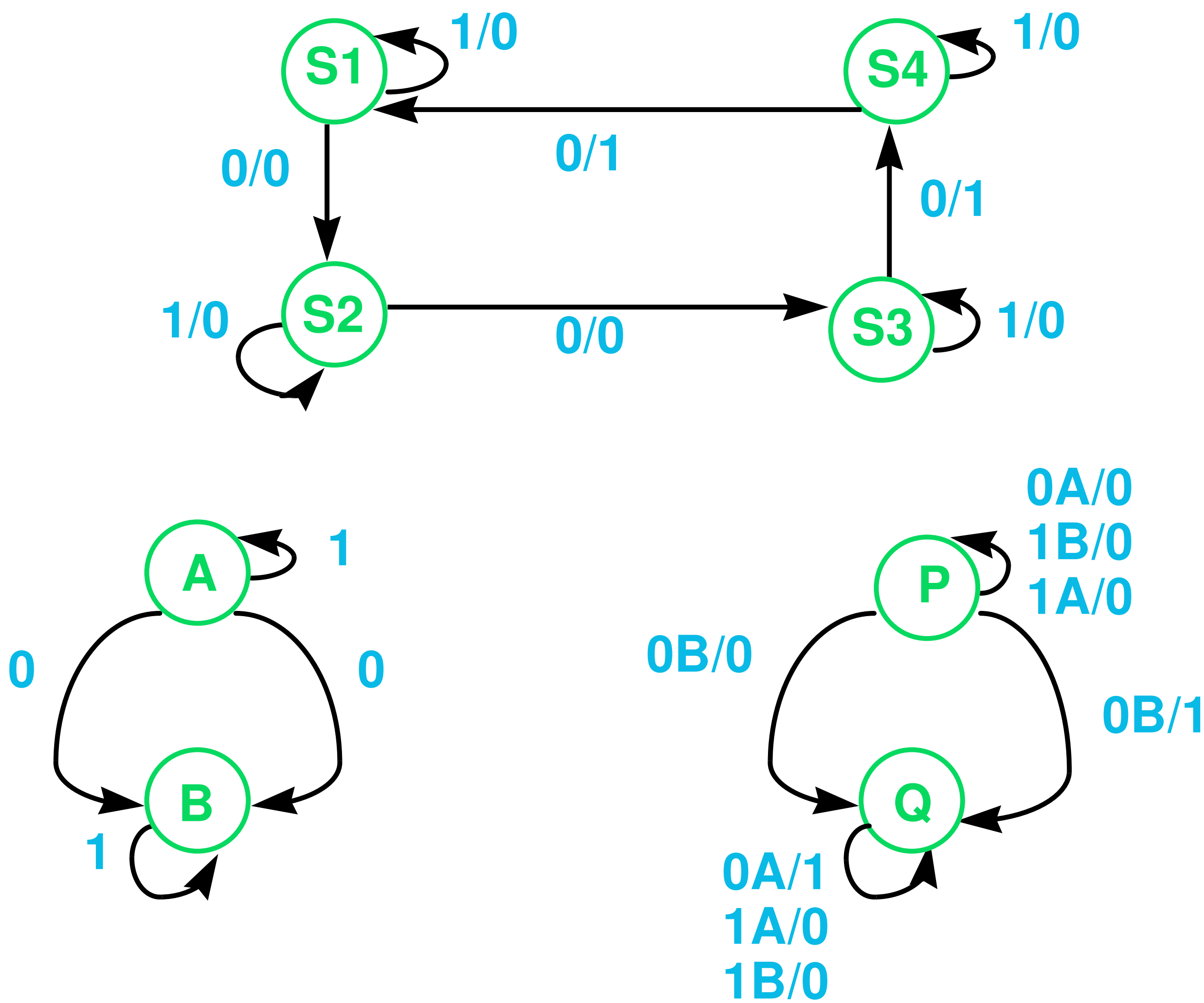
General decomposition: Both machines "know" state of the other.

Parallel Decomposition

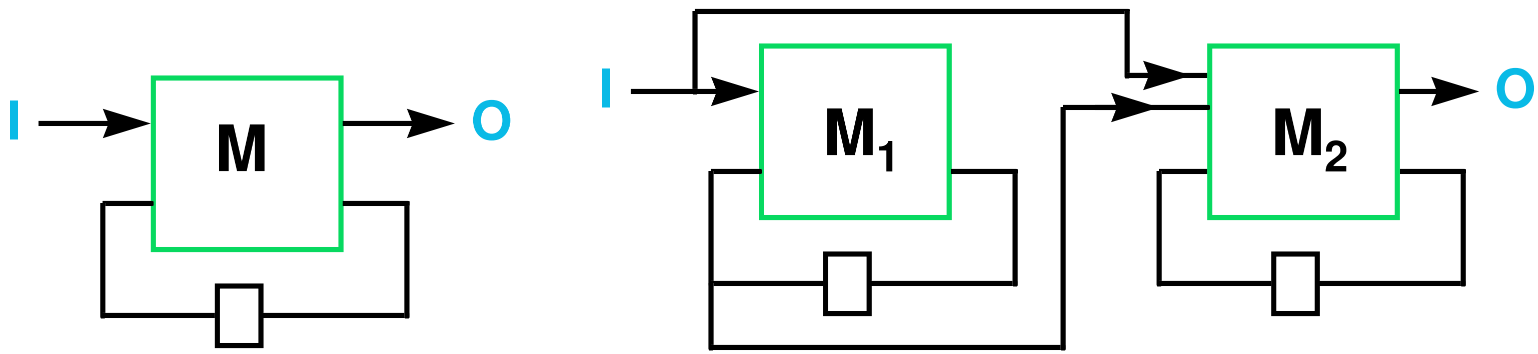




Cascade Decomposition



Cascade Decomposition Algorithm



Independent (M_1) and dependent (M_2) component.

Given the original machine M and an independent component M_1 that is arbitrary, we can always construct a dependent component M_2 to realize the behavior of M.

Choice of independent component is critical.

Decomposition Algorithm - II

We have to construct the dependent machine M_2 in the cascade $M_1 \rightarrow M_2$.

Construct a composite graph that is the parallel realization of M and M_1.

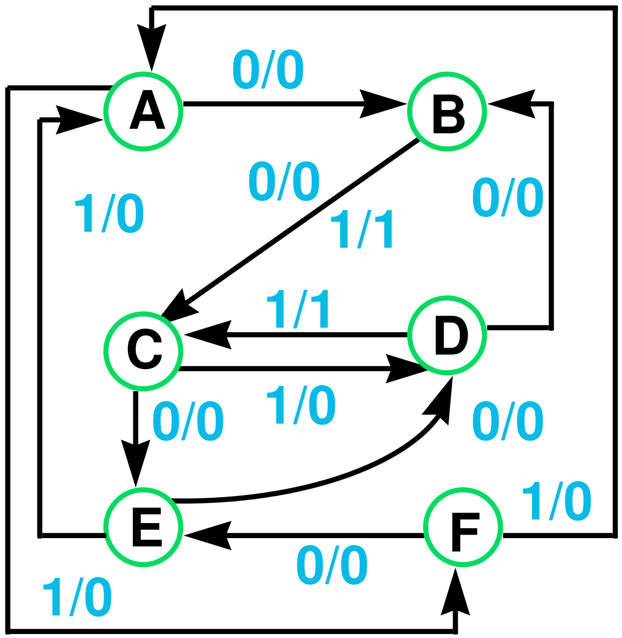
Inspect the composite graph to determine the number of states in M_2.

Choose mapping of states in M_1, M_2 to states in M.

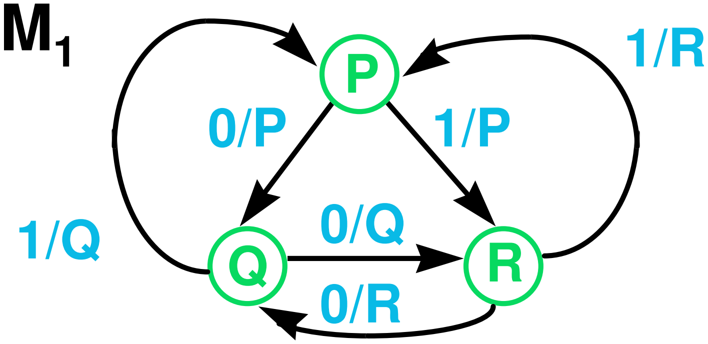
Construct M_2 while maintaining terminal behavior.

Construct Parallel Machine

M

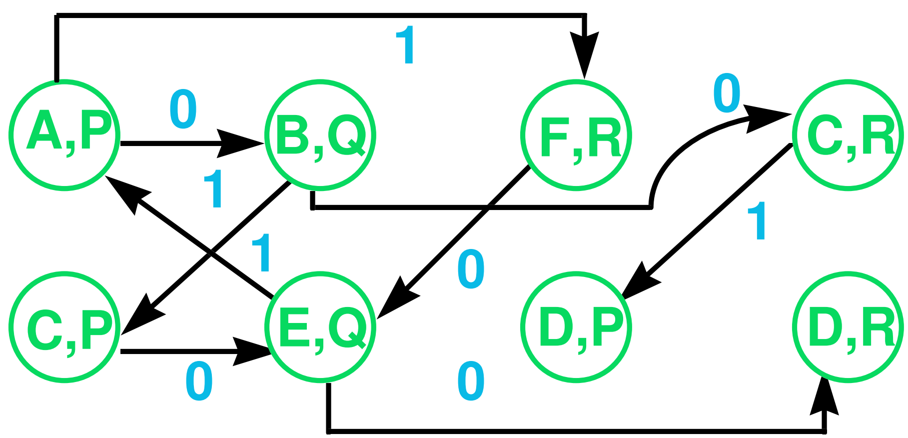


M_1



(Output is present state)

Parallel machine assuming **A** and **P** are starting states.



could have had 18 states

A,Q ?

Ambiguity Vector

The parallel machine tells us what states M can be in when M_1 is in each of its states.

States in parallel machine are:

AP BQ FR CR CP EQ DP DR

M_1	M	
P	A,C,D	AP, CP, DP
Q	B,E	BQ, EQ
R	C,D,F	CR, DR, FR

"Knowing" the state of M_1 we can specify state of M to within 3 states

(ACD) (BE) (CDF) is called the ambiguity vector

Resolving Ambiguity

The dependent component must be provided with 3 states J, K, L since ambiguity vector has term(s) of size 3.

Choose a way of assigning these three states J, K, L to the members of the ambiguities

(ACD) (BE) (CDF)

One Choice

	P	Q	R
J	A	B	C
K	C	E	D
L	D		F

Another Choice

	P	Q	R
J	A	B	F
K	C	E	C
L	D		D

Here each state of M has a unique state in M_2 (not always possible)

Construct Dependent Component

	0	1
A	B 0	F 0
B	C 0	C 1
C	E 0	D 0
D	B 0	C 1
E	D 0	A 0
F	E 0	A 0

M

	0	1
P	Q	R
Q	R	P
R	Q	P

M_1

	P	Q	R
J	A	B	C
K	C	E	D
L	D		F

	x = 0			x = 1		
	P	Q	R	P	Q	R
J						
K						
L				J 1		

M_2

M_1 is in P, M_2 is in L
Input is 1
M_1 = P $\Rightarrow$ L represents original state D

M = D and receives 1, it moves to C and produces 1

M_1 goes to R $\Rightarrow$ M_2 has to go to J to represent D

Dependent Component - 2

	x = 0 P	x = 0 Q	x = 0 R	x = 1 P	x = 1 Q	x = 1 R
J	J 0	J 0	K 0	L 0	K 1	L 0
K	K 0	K 0	J 0	K 0	J 0	K 1
L	J 0		K 0	J 1		J 0

M_1 is in Q, M_2 is in K, input is 0

When M_1 is in Q, state K of M_2 represents E in M

M in E receives 0, it goes to D with output 0

M_1 receives 0, it goes to R from Q

$\therefore$ M_2 has to go to K to represent D in the next cycle

Finding Independent Components

The ambiguity vector associated with an independent component is a preserved cover [(ACD) (BE) (CDF)]

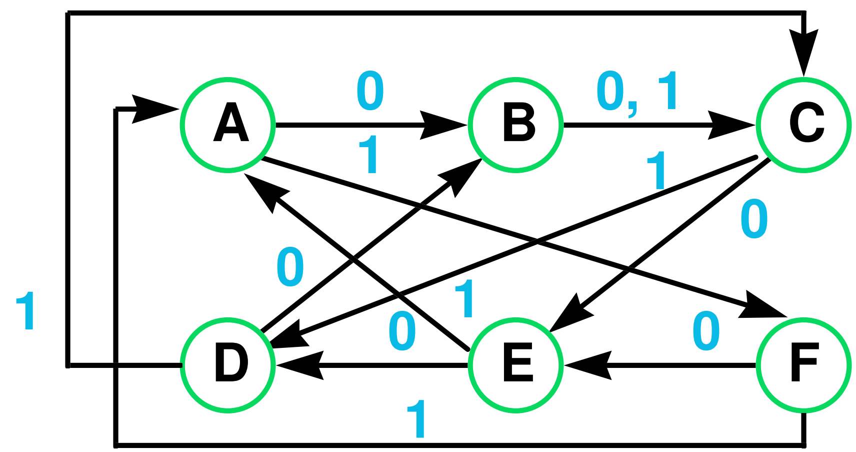
<u>Definition:</u> A preserved cover for a machine M is m blocks of state $(q_{11} \cdot\cdot\ q_{1N1})$ $(q_{21}, q_{22} \cdot\cdot\ q_{2N2})$ $\cdot\cdot$ $(q_{m1}, \cdot\cdot\ q_{mNm})$ such that

(1) Each state in M appears in the cover

(2) No block of states is contained in any other block and

(3) For each input value σ the σ -successors of the members of any block should all be contained within some common block

Independent Components - 2



Not concerned with outputs, but merely with connectivity
(ACD) (BE) (CDF) is a preserved cover

1-successors of ACD = FDC (CDF)
0-successors of ACD = BEB (BE)
1-successors of BE = CA (ACD)
0-successors of BE = CD (ACD)
1-successors of CDF = DCA (ACD)
0-successors of CDF = EBE (BE)

Covers and Decompositions

The choice of a preserved cover characterizes the major features of a decomposition.

- Number of blocks in the cover determines the number of states in the independent component.

- Successor relations among the blocks determine the successor relations among the states of the independent component.

- The maximum number of states in any block determines the number of states in the dependent component (which has to resolve the ambiguity).

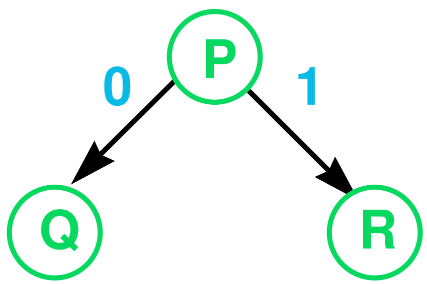
Preserved Covers

	0	1
A	B	F
B	C	C
C	E	D
D	B	C
E	D	A
F	E	A

(ACD) (BE) (CDF)

Assign one state of M_1 to each block in the cover

P ← (ACD)   Q ← (BE)   R ← (CDF)



0-successor of (ACD) is (BE)

$\therefore P \xrightarrow{0} Q$

1-successor of (ACD) is (FDC)

$\therefore P \xrightarrow{1} R$

Covers in a Machine

Machine usually has many preserved covers

	0	1
A	B	F
B	C	C
C	E	D
D	B	C
E	D	A
F	E	A

(ABCDEF)
(ACDF)(BCDE)
(ACD)(BE)(CDF)
(AC)(BE)(CD)(DF)
(AD)(B)(CF)(E)
(A)(B)(C)(D)(E)(F)

(ABCDEF) and (A)(B)(C)(D)(E)(F) are trivial covers.

Using (ACDF)(BCDE) will imply that independent component has two states and dependent component has four states.

Universal Covers

Any machine has $n - 2$ non-trivial preserved covers (n is the # states)

Pick a value m in the range $1 < m < n$. List all possible blocks containing exactly m states.

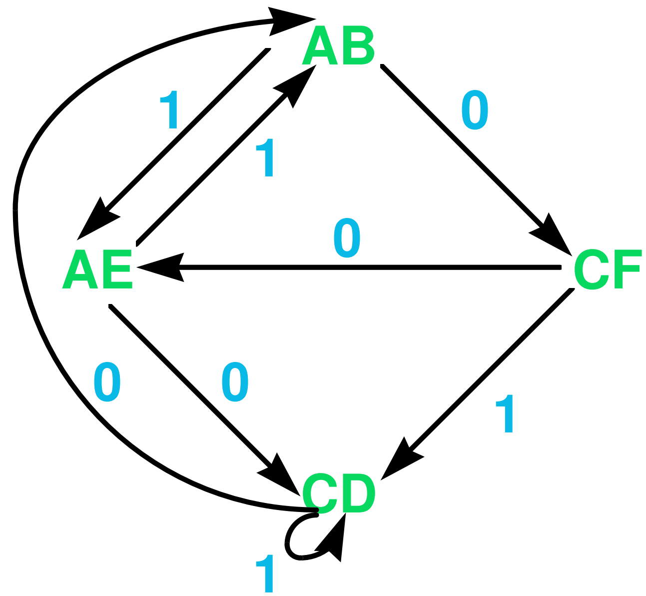
Because each successor of a set of m states must be a set of m or fewer states, the resulting cover is necessarily preserved (since all blocks with m states are in the cover)

Such covers are called universal covers ($n = 6, m = 2$; number of blocks = 15)

Finding Preserved Covers

Pick m = 2, i.e., restrict to 2-state blocks in cover

	0	1
A	C	A
B	F	E
C	A	D
D	B	C
E	D	B
F	E	C



Pick a trial block (AB)

Form a graph with (AB) as root

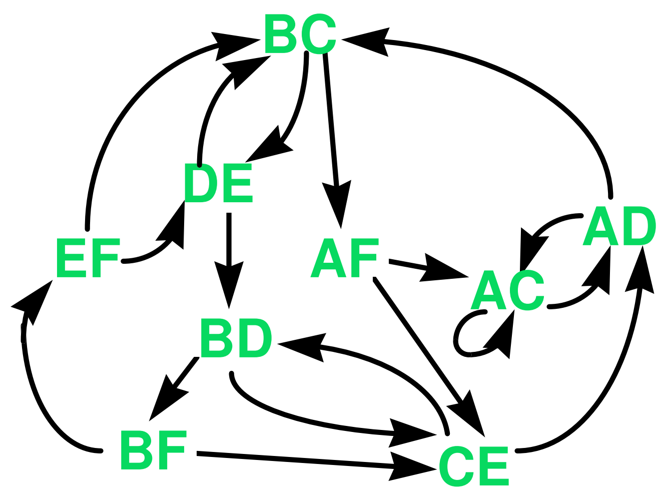
Graph represents a preserved cover (AB)(AE)(CF)(CD)

graph has to be "closed" in order to be a cover

Trial Blocks

	0	1
A	C	A
B	F	E
C	A	D
D	B	C
E	D	B
F	E	C

Pick (AC) as a trial block



disaster ⇒ too many blocks in cover

Selection of trial block is critical.

Successor Graphs

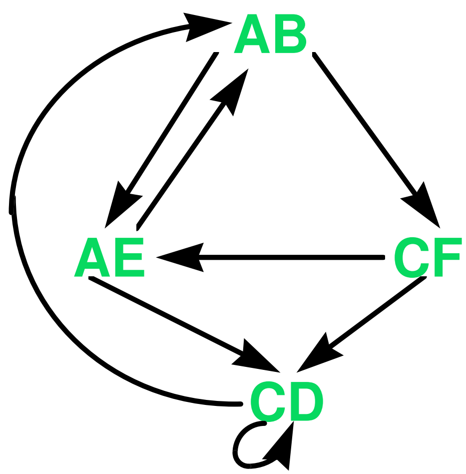
Orderly way of determining which trial blocks lead to useful covers is to form a successor graph for all 2-state (or 3-state) blocks.

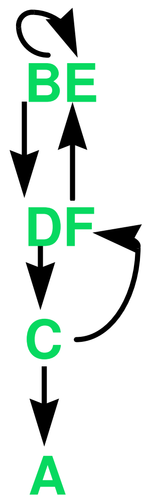
We need not label the transitions, since we only want to know what blocks are to be included.

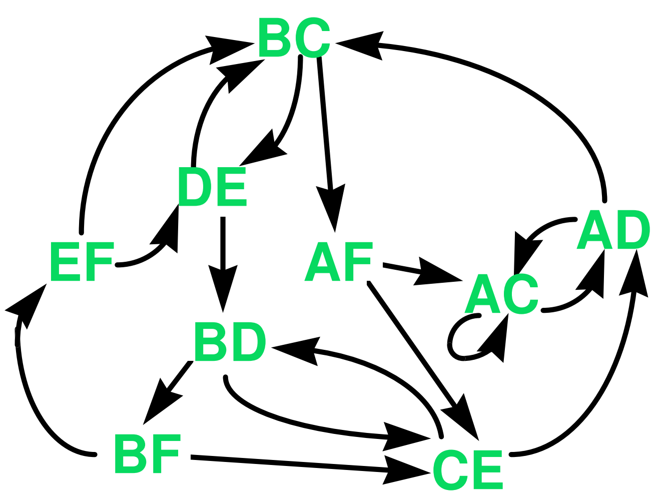
Look at graph and search for disjoint clusters that are small.

Successor Graphs - 2

	0	1
A	C	A
B	F	E
C	A	D
D	B	C
E	D	B
F	E	C







Restricting oneself to covers with exactly m states

⇒ may miss useful covers

No completely satisfactory solution

Summary

Given state graph minimize number of states

Attempt to find parallel or cascade decomposition

Apply heuristic encoding methods to original or decomposed machines.