Capabilities, Minimization, and Transformation of Sequential Machines

Zvi Kohavi and Niraj K. Jha

Finite-state Model

Deterministic machines: next state $S(t+1)$ determined uniquely by present state $S(t)$ and present input $x(t)$

$$S(t+1) = \delta\{S(t), x(t)\}$$

- δ: state transition function

Output function λ:

$$z(t) = \lambda\{S(t), x(t)\}: \text{ Mealy machine}$$
$$z(t) = \lambda\{S(t)\}: \text{ Moore machine}$$

Synchronous sequential machine M:

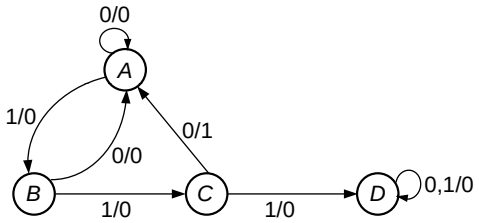
$$M = \{I, O, S, \delta, \lambda\}$$

- I: set of input symbols
- O: set of output symbols
- S: set of states
- δ: $I \times S \rightarrow S$ is the state transition function
- λ: $I \times S \rightarrow O$ for Mealy machines
- λ: $S \rightarrow O$ for Moore machines

Input-output Transformations

Example: four-state machine *M* with one input and one output variable

$S = \{A,B,C,D\}$, $I = \{0,1\}$, $O = \{0,1\}$

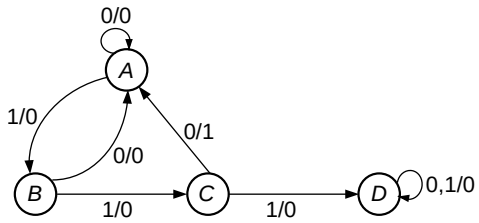


- Suppose initial state is *A*: *M* transforms input sequence 110 to output sequence 001, and 01100 to 00010
- If last output symbol is 1 (0): the corresponding input sequence is said to be accepted (rejected) by *M*
  - 110 is accepted; 01100 is rejected
- If input sequence *X* takes machine from state S_i to S_j: S_j is said to be the *X*-successor of S_i
- *B*: 1-successor of *A*
- (*AD*): 10-successor of (*BC*)

Terminal State

A state is called terminal if:

- Corresponding vertex is a sink vertex: no outgoing arcs emanating from it terminate in other vertices
- Corresponding vertex is a source vertex: no arcs emanating from other vertices terminate in it



- *D*: a sink vertex

Strongly connected machine: for every pair of states S_i, S_j of machine *M*, there exists an input sequence which takes *M* from S_i *to* S_j

- Any machine that has a terminal state is not strongly connected

Capabilities and Limitations of FSMs

Apply a string of m 1's to an n-state FSM, $m > n$: some state must be revisited

- Output sequence becomes periodic, whose period cannot exceed n

Example: Design a machine that receives a long sequence of 1's and produces output symbol 1 when and only when the number of input symbols received so far is $k(k+1)/2$, for $k = 1, 2, 3, \ldots$, i.e.

Input = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 …

Output = 1 0 1 0 0 1 0 0 0 1 0 0 0 0 1 …

- Since the output sequence does not become periodic: no FSM can produce such an infinite sequence

Capabilities and Limitations (Contd.)

Example: No FSM with a fixed number of states can multiply two arbitrarily large numbers

- Suppose there exists an n-state machine capable of serially multiplying any two binary numbers
- Select the two numbers to be $2^p \times 2^p = 2^{2p}$, where $p > n$
- Input values are fed to the machine serially, LSB first
- 2^p: 1 followed by p 0's; 2^{2p}: 1 followed by $2p$ 0's
- Since $p > n$, the machine must have been at one of the states twice during t_{p+1} and t_{2p}
  - Thus, the output must become periodic and the period is smaller than p: hence, it will never produce the 1 output symbol

t_{2p+1}	t_{2p}	$\cdots$	t_{p+1}	t_p	$\cdots$	t_2	t_1	= time
			1	0	$\cdots$	0	0	= first number
			1	0	$\cdots$	0	0	= second number
1	0	$\cdots$	0	0	$\cdots$	0	0	= product

State Equivalence and Machine Minimization

k-distinguishable: Two states S_i and S_j of machine *M* are distinguishable if and only if there exists at least one finite input sequence which, when applied to *M*, causes different output sequences, depending on whether S_i or S_j is the initial state

- Sequence which distinguishes these states: distinguishing sequence of pair (S_i,S_j)
- If there exists a distinguishing sequence of length *k* for (S_i,S_j): S_i and S_j are said to be *k*-distinguishable

Example: In machine M_1

- (*A*,*B*): 1-distinguishable
- (*A*,*E*): 3-distinguishable since the minimum sized sequence that distinguishes *A* and *E* is 111

	NS, z	
PS	$x=0$	$x=1$
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	$B,0$	$C,0$

Machine M_1

State Equivalence (Contd.)

k-equivalence: States that are not *k*-distinguishable are *k*-equivalent

- States *A* and *E* of M_1 are 2-equivalent
- States that are *k*-equivalent are also *r*-equivalent, for all $r < k$
- States that are *k*-equivalent for all *k* are said to be equivalent

States S_i and S_j of machine *M* are equivalent (indicated by $S_i = S_j$): 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

- Clearly, if $S_i = S_j$ and $S_j = S_k$, then $S_i = S_k$
  - Thus, state equivalence is an equivalence relation
  - The set of states of the machine can be partitioned into disjoint subsets, known as equivalence classes
- This definition can be generalized to the case: where S_i is a possible initial state of machine M_1, while S_j is an initial state of machine M_2, where M_1 and M_2 have the same input alphabet

State Minimization Procedure

If S_i and S_j are equivalent states, their corresponding X-successors, for all X, are also equivalent: since otherwise it would be trivial to construct a distinguishing sequence for (S_i,S_j) by first applying an input sequence that transfers the machine to the distinguishable successors of S_i and S_j

Machine M_1

PS	NS, z $x=0$	$x=1$
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	$B,0$	$C,0$

Example: For machine M_1

- P_0, P_1: 0-distinguishable, 1-distinguishable
- P_2: two states placed in the same block if and only if they are in the same block of P_1, and for each possible I_i, their I_i-successor is also contained in a common block of P_1
  - 0- and 1-successor of (*ACE*): (*CE*), (*BDF*)
    - » Since both are contained in common blocks of P_1: states in (*ACE*) are 2-equivalent
  - 1-successor of (*BDF*): (*DBC*)
    - » Since (*DB*) and (*C*) are not contained in a single block of P_1: (*BDF*) must be split into (*BD*) and (*F*), and so on
- Since $P_3 = P_4$: P_3 is the equivalence partition

$$P_0 = (ABCDEF)$$
$$P_1 = (ACE)(BDF)$$
$$P_2 = (ACE)(BD)(F)$$
$$P_3 = (AC)(E)(BD)(F)$$
$$P_4 = (AC)(E)(BD)(F)$$

Theorems

Theorem 10.1: The equivalence partition is unique

Proof: Suppose there exist two partitions, P_a and P_b, and that $P_a = P_b$.

- Then there exist two states, S_i and S_j, which are in the same block of one partition and not in the same block of the other
- Since S_i and S_j are in different blocks of (say) P_b, there exists at least one input sequence which distinguishes S_i from S_j and, therefore, cannot be in the same block of P_a

Theorem 10.2: If two states, S_i and S_j, of machine M of n states are distinguishable, then they are distinguishable by a sequence of length n-1 or smaller

Proof: P_1 contains at least two blocks, else M is reducible to a combinational circuit with a single state

- At each step, partition P_{k+1} is smaller than or equal to P_k
  - If P_{k+1} is smaller than P_k, then it contains at least one more block than P_k
  - However, since the number of blocks is limited to n, at most n-1 partitions can be generated
  - Thus, if S_i and S_j are distinguishable, they are distinguishable by a sequence of length n-1 or smaller

Machine Equivalence

Two machines, M_1 and M_2, are said to be equivalent if and only if, for every state in M_1, there is a corresponding state in M_2, and vice versa

- The machine that contains no equivalence states and is equivalent to M is called the minimal, or reduced, form of M

Example:

Machine M_1

	NS, z	
PS	$x=0$	$x=1$
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	$B,0$	$C,0$

Machine M_1*

	NS, z	
PS	$x=0$	$x=1$
α	$\beta,0$	$\gamma,1$
β	$\alpha,0$	$\delta,1$
γ	$\delta,0$	$\gamma,0$
δ	$\gamma,0$	$\alpha,0$

$$
\begin{aligned}
P_0 &= (ABCDEF) \\
P_1 &= (ACE)(BDF) \\
P_2 &= (ACE)(BD)(F) \\
P_3 &= (AC)(E)(BD)(F) \\
P_4 &= (AC)(E)(BD)(F)
\end{aligned}
$$

Machine Equivalence (Contd.)

Example:

Machine M_2

	NS, z	
PS	$x=0$	$x=1$
A	$E,0$	$C,0$
B	$C,0$	$A,0$
C	$B,0$	$G,0$
D	$G,0$	$A,0$
E	$F,1$	$B,0$
F	$E,0$	$D,0$
G	$D,0$	$G,0$

Machine M_2*

	NS, z	
PS	$x=0$	$x=1$
$(A) \rightarrow \alpha$	$\epsilon,0$	$\delta,0$
$(F) \rightarrow \beta$	$\epsilon,0$	$\gamma,0$
$(BD) \rightarrow \gamma$	$\delta,0$	$\alpha,0$
$(CG) \rightarrow \delta$	$\gamma,0$	$\delta,0$
$(E) \rightarrow \epsilon$	$\beta,1$	$\gamma,0$

$$
\begin{aligned}
P_0 &= (ABCDEFG) \\
P_1 &= (ABCDFG)(E) \\
P_2 &= (AF)(BCDG)(E) \\
P_3 &= (AF)(BD)(CG)(E) \\
P_4 &= (A)(F)(BD)(CG)(E) \\
P_5 &= (A)(F)(BD)(CG)(E)
\end{aligned}
$$

Isomorphic Machines

If one machine can be obtained from another by relabeling its states: the machines are said to be isomorphic to each other

- To every machine *M*, there corresponds a minimal machine *M** which is equivalent to *M* and is unique up to isomorphism

Example: Machine in standard form

	NS, z	
PS	$x=0$	$x=1$
$(A) \rightarrow \alpha$	$\epsilon, 0$	$\delta, 0$
$(F) \rightarrow \beta$	$\epsilon, 0$	$\gamma, 0$
$(BD) \rightarrow \gamma$	$\delta, 0$	$\alpha, 0$
$(CG) \rightarrow \delta$	$\gamma, 0$	$\delta, 0$
$(E) \rightarrow \epsilon$	$\beta, 1$	$\gamma, 0$

	NS, z	
PS	$x=0$	$x=1$
$\alpha \rightarrow A$	$B, 0$	$C, 0$
$\epsilon \rightarrow B$	$D, 1$	$E, 0$
$\delta \rightarrow C$	$E, 0$	$C, 0$
$\beta \rightarrow D$	$B, 0$	$E, 0$
$\gamma \rightarrow E$	$C, 0$	$A, 0$

Simplification of Incompletely Specified Machines

Incompletely specified machine: in which the next state or output symbol is left unspecified

- When a state transition is unspecified: the future behavior of the machine may become unpredictable
  - Assume that the input sequences are such that no unspecified next state is encountered, except possibly at the final step
    - » Such an input sequence is said to be applicable to the starting state of the machine

Example: Machine M_3

	NS, z	
PS	$x=0$	$x=1$
A	$B, 1$	—
B	–, 0	$C, 0$
C	$A, 1$	$B, 0$

Equivalent description

	NS, z	
PS	$x=0$	$x=1$
A	$B, 1$	T, –
B	$T, 0$	$C, 0$
C	$A, 1$	$B, 0$
T	T, –	T, –

State/Machine Covering

State S_i of M_1 is said to cover, or contain, state S_j of M_2 if and only if every input sequence applicable to S_j is also applicable to S_i

- Its application to both M_1 and M_2 when they are in S_i and S_j, respectively, results in identical output sequences whenever the output symbols of M_2 are specified

Machine M_1 is said to cover machine M_2 if and only if, for every state S_j in M_2, there is a corresponding state S_i in M_1 such that S_i covers S_j

Example:

PS	NS, z	
	$x=0$	$x=1$
A	$B,1$	$T,-$
B	$T,0$	$C,0$
C	$A,1$	$B,0$
T	$T,-$	$T,-$

covers

PS	NS, z	
	$x=0$	$x=1$
A	$B,1$	—
B	$-,0$	$C,0$
C	$A,1$	$B,0$

Compatible States

Two states, S_i and S_j, of machine M are compatible if and only if, for every input sequence applicable to both S_i and S_j, the same output sequence will be produced whenever both output symbols are specified regardless of whether S_i or S_j is the initial state

- Hence, S_i and S_j are compatible: if and only if their output symbols are not conflicting and their I_i-successors, for every I_i for which both are specified, are either the same or also compatible
- A set of states (S_i, S_j, S_k, …) is called a compatible: if all its members are compatible
- A compatible C_i is said to be larger than, or to cover, another compatible C_j: if and only if every state contained in C_j is also contained in C_i
- A compatible is maximal: if is not covered by any other compatible
- In case of the incompletely specified machine: the analog to the equivalence relation is the compatibility relation

Nonuniqueness of Reduced Machines

Example: Machine M_4

PS	NS, z	
	$x-0$	$x-1$
A	$C,1$	$E,-$
B	$C,-$	$E,1$
C	$B,0$	$A,1$
D	$D,0$	$E,1$
E	$D,1$	$A,0$

- If we replace both dashes by 1's: *A* and *B* become equivalent
- If we replace both dashes by 0's: States *A* and *E* become equivalent; also, *B*, *C*, and *D* become equivalent

PS	NS, z	
	$x=0$	$x=1$
A	$C,1$	$E,1$
C	$A,0$	$A,1$
D	$D,0$	$E,1$
E	$D,1$	$A,0$

PS	NS, z	
	$x=0$	$x=1$
$(AE) \rightarrow \alpha$	$\beta,1$	$\alpha,0$
$(BCD) \rightarrow \beta$	$\beta,0$	$\alpha,1$

- Both reduced machines cover M_4: thus reduced machines are nonunique
- States *A* and *B* of M_4 are compatible, and if *C* and *D* are also compatible, so are *A* and *E*. However, *B* and *E* are 1-distinguishable, hence incompatible: thus compatibility relation is not an equivalence relation

Nonuniqueness of Minimal Machines

Example: Machine M_5

PS	NS, z	
	$x=0$	$x=1$
A	$A,0$	$C,0$
B	$B,0$	$B,-$
C	$B,0$	$A,1$

Augmented machine

PS	NS, z	
	$x=0$	$x=1$
A	$A,0$	$C,0$
B'	$B',0$	$B'',-$
B''	$B^+,0$	$B',-$
C	$B^+,0$	$A,1$

- Two minimal machines corresponding to M_5

PS	NS, z	
	$x=0$	$x=1$
$(AB') \rightarrow \alpha$	$\alpha,0$	$\beta,0$
$(B''C) \rightarrow \beta$	$\alpha,0$	$\alpha,1$

(a) Setting $B^+ = B'$.

PS	NS, z	
	$x=0$	$x=1$
$(AB') \rightarrow \alpha$	$\alpha,0$	$\beta,0$
$(B''C) \rightarrow \beta$	$\beta,0$	$\alpha,1$

(b) Setting $B^+ = B''$.

- While the equivalence partition consists of disjoint blocks, the subsets of compatibles may be overlapping

Compatible/Implied Pair

Let the I_k-successors of S_i and S_j be S_p and S_q, respectively: then (S_pS_q) is said to be the implied pair of the compatible pair (S_iS_j)

Example: (CF) is the implied pair of the compatible pair (AC)

	NS, z			
PS	I_1	I_2	I_3	I_4
A	—	$C,1$	$E,1$	$B,1$
B	$E,0$	—	—	—
C	$F,0$	$F,1$	—	—
D	—	—	$B,1$	—
E	—	$F,0$	$A,0$	$D,1$
F	$C,0$	—	$B,0$	$C,1$

Merger Graph

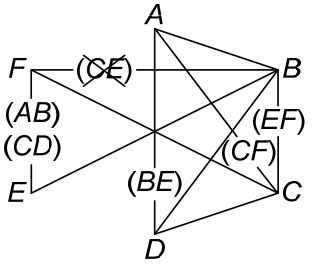
Merger graph of an n-state machine M is an undirected graph defined as follows:

1. It consists of n vertices, each of which corresponds to a state of M
2. For each pair of states (S_iS_j) in M, whose next-state and output entries are not conflicting, an undirected arc is drawn between vertices S_i and S_j
3. If, for a pair of states (S_iS_j), the corresponding output symbols under all input symbols are not conflicting, but the successors are not the same, an interrupted arc is drawn between S_i and S_j, and the implied pairs are entered in the space

Example: Machine M_6

	NS, z			
PS	I_1	I_2	I_3	I_4
A	—	$C,1$	$E,1$	$B,1$
B	$E,0$	—	—	—
C	$F,0$	$F,1$	—	—
D	—	—	$B,1$	—
E	—	$F,0$	$A,0$	$D,1$
F	$C,0$	—	$B,0$	$C,1$

Merger graph



Nine compatible pairs: (AB), (AC), (AD), (BC), (BD), (BE), (CD), (CF), (EF)

Set of maximal compatibles: $\{(ABCD), (BE), (CF), (EF)\}$ – complete polygons

Closed Sets of Compatibles

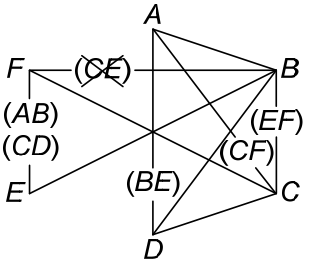
A set of compatibles for machine *M* is said to be closed if: for every compatible contained in the set, all its implied compatibles are also contained in the set

- A closed set of compatibles, which contains all states of *M*, is called a closed covering

Example: {(*ABCD*), (*EF*)} has the minimal number of compatibles covering all states of M_6

- It defines a lower bound on the number of states in the minimal machine that covers M_6
- However, if we select maximal compatible (*ABCD*) to be a state in the reduced machine: then its I_2- and I_3-successors, (*CF*) and (*BE*), must also be selected – since these are not in the above set, set {(*ABCD*), (*EF*)} cannot be used to define the states of a minimal machine for M_6

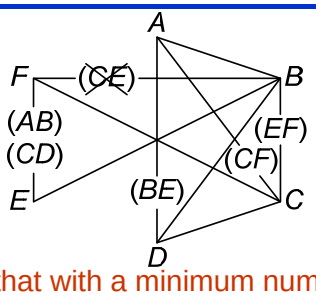
	NS, z			
PS	I_1	I_2	I_3	I_4
A	—	$C,1$	$E,1$	$B,1$
B	$E,0$	—	—	—
C	$F,0$	$F,1$	—	—
D	—	—	$B,1$	—
E	—	$F,0$	$A,0$	$D,1$
F	$C,0$	—	$B,0$	$C,1$



Closed Covering

Example (contd.): Closed covers:

- {(*AD*), (*BE*), (*CF*)}
- {(*AB*), (*CD*), (*EF*)}



Closed covering is not unique

- Aim is to find a closed covering that with a minimum number of compatibles
- Set of all maximal compatibles: clearly a closed covering
  - This defines an upper bound on the number of states in the machine that covers the original one:
    - » The upper bound is meaningless when the number of maximal compatibles is larger than the number of states in the original machine
- For the example: the lower bound is 2 and upper bound 4
  - Thus, a closed covering with three compatibles defines a minimal machine

	NS, z			
PS	I_1	I_2	I_3	I_4
$(AB) \rightarrow \alpha$	$\gamma,0$	$\beta,1$	$\gamma,1$	$\alpha,1$
$(CD) \rightarrow \beta$	$\gamma,0$	$\gamma,1$	$\alpha,1$	—
$(EF) \rightarrow \gamma$	$\beta,0$	$\gamma,0$	$\alpha,0$	$\beta,1$

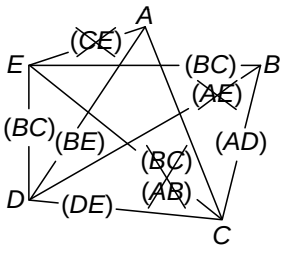
Compatibility Graph

Compatibility graph: a directed graph whose vertices correspond to all compatible pairs, and an arc leads from (S_iS_j) to (S_pS_q) if and only if (S_iS_j) implies (S_pS_q)

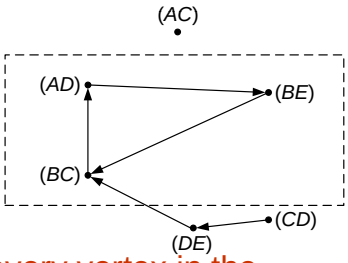
Example: Machine M_7

PS	NS, z I_1	I_2	I_3	I_4
A	—	—	$E,1$	—
B	$C,0$	$A,1$	$B,0$	—
C	$C,0$	$D,1$	—	$A,0$
D	—	$E,1$	$B,-$	—
E	$B,0$	—	$C,-$	$B,0$

Merger graph



Compatibility graph



A subgraph of the compatibility graph is closed: if, for every vertex in the subgraph, all outgoing arcs and their terminating vertices also belong to it

- If every state of the machine is covered by at least one vertex of the subgraph: then the subgraph forms a closed covering
  - {(*BC*), (*AD*), (*BE*)}; {(*AC*), (*BC*), (*AD*), (*BE*)}; {(*DE*), (*BC*), (*AD*), (*BE*)}

Minimal machine

PS	NS, z I_1	I_2	I_3	I_4
$(AD) \rightarrow \alpha$	—	$\gamma,1$	$\gamma,1$	—
$(BC) \rightarrow \beta$	$\beta,0$	$\alpha,1$	$\beta/\gamma,0$	$\alpha,0$
$(BE) \rightarrow \gamma$	$\beta,0$	$\alpha,1$	$\beta,0$	$\beta/\gamma,0$

Merger Table

Merger table: more convenient alternative to the merger graph

Example: Machine M_8

PS	NS, z I_1	I_2
A	$E,0$	$B,0$
B	$F,0$	$A,0$
C	$E,-$	$C,0$
D	$F,1$	$D,0$
E	$C,1$	$C,0$
F	$D,-$	$B,0$

Merger table

	A	B	C	D	E
B	EF				
C	BC	AC, EF			
D	×	×	EF		
E	×	×	✓	CD, CF	
F	DE	~~AB~~, ~~DF~~	BC, DE	~~BD~~	BC, CD

Finding the Set of Maximal Compatibles

Tabular counterpart to finding complete polygons in the merger graph:

1. Start in the rightmost column of the merger table and proceed left until a column containing a compatible pair is encountered – list all compatible pairs in that column
2. Proceed left to the next column containing at least one compatible pair. If the state to which this column corresponds is compatible with all members of some previously determined compatible, add this state to that compatible to form a larger compatible. If the state is not compatible with all members, but is compatible with some, form a new compatible that includes those members and the state in question. Next, list all compatible pairs not included in any previously derived compatible
3. Repeat step 2 until all columns have been considered. The final set of compatibles constitutes the set of maximal compatibles

Maximal Compatibles (Contd.)

Example:

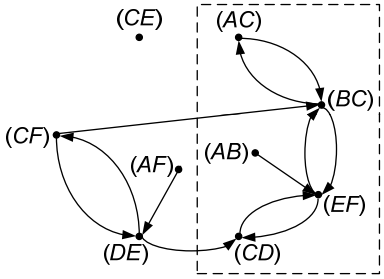
	A	B	C	D	E
B	*EF*				
C	*BC*	*AC*, *EF*			
D	×	×	*EF*		
E	×	×	✓	*CD*, *CF*	
F	*DE*	~~*AB*~~, *DF*	*BC*, *DE*	~~*BD*~~	*BC*, *CD*

Column *E*: (*EF*)
Column *D*: (*EF*), (*DE*)
Column *C*: (*CEF*), (*CDE*)
Column *B*: (*CEF*), (*CDE*), (*BC*)
Column *A*: (*CEF*), (*CDE*), (*ABC*), (*ACF*)

- Set of maximal compatibles indicates that M_8 can be covered by a four-state machine, but not by a two-state machine

Maximal Compatibles (Contd.)

Example (contd.): Compatibility graph



- Add (*AB*) to closed subgraph {(*AC*), (*BC*), (*EF*), (*CD*)}
  - Reduces to the following closed covering: {(*ABC*), (*CD*), (*EF*)}
- Minimal machine:

PS	NS, z	
	I_1	I_2
$(ABC) \rightarrow \alpha$	$\gamma, 0$	$\alpha, 0$
$(CD) \rightarrow \beta$	$\gamma, 1$	$\beta, 0$
$(EF) \rightarrow \gamma$	$\beta, 1$	$\alpha, 0$