FSM minimisation

Limitations of FSMs

Computing $2^p \times 2^p$

- Result: 2^{2p} i.e. 1 followed by $2p$ zeroes, length $2p+1$
- Let the FSM have n states
- Input length: $p+1$
- As the numbers are input (serially) $2p+1$ zeroes should be output
- Next, in $p-1$ zeroes should be output and finally a 1
- Let $p > n$, the m/c must loop through the states to generate the $p-1$ zeroes
- The required 1 will never be generated

State equivalence

Basic notions

Distinguishable states
: S_i and S_j of M are distinguishable if and only if a finite input sequence applied to M produce distinct output sequence, depending of whether M is in S_i or S_j

Distinguishing sequence for S_i and S_j
: A sequence that allows S_i and S_j to be distinguished

k-distinguishable
: S_i and S_j are k-distinguishable is there exists a sequence of length k to distinguish S_i and S_j

k-equivalent
: No sequence of length k to distinguish S_i and S_j

Equivalent states
: S_i and S_j are equivalent if they are k-equivalent for all k; for n state m/c sufficient if k-equivalent for all $k \leq n-1$ (in view of the limited memory of FSMs)

Minimisation of completely specified FSMs

Sample m/c and partition table

FSM specification

PS	NS, output	
	$x=0$	$x=1$
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

Equivalence partition table

k	Partitions
P_0	(ABCDEF)
P_1	
P_2	
P_3	
P_4	

Equivalence partition is unique

- Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two distinct equivalence partitions
- There there must be two states s_i and s_j in the same partition of $\mathcal{P}_1$ but in different partitions of $\mathcal{P}_2$
- Since those are in different partitions in $\mathcal{P}_1$, those are distinguishable and cannot be in the same equivalence partition of $\mathcal{P}_2$
- Thus, the two equivalence partitions cannot be distinct
- By corollary, the equivalence partition $\mathcal{P}$ is minimum — a distinct equivalence partition $\mathcal{P}'$ will not exist

---

Minimisation of incompletely specified FSMs

Sample incompletely specified FSMs

FSM specification: M_A

PS	NS, output	
	$x=0$	$x=1$
A	C,1	E,—
B	C,—	E,1
C	B,0	A,1
D	D,0	E,1
E	D,1	E,0

FSM specification: M_B

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

FSM specification: M_C

PS	NS, output	
	$x=0$	$x=1$
A	B,1	T,—
B	T,0	C,0
C	A,1	B,0
T	T,—	T,—

FSM specification: M_D

PS	NS, output			
	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

FSM specification: M_E

PS	NS, output	
	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

FSM specification: M_F

PS	NS, output			
	I_1	I_2	I_3	I_4
A	—	D,0	D,0	C,—
B	A,1	—	—	D,—
C	B,1	C,0	E,0	—
D	E,1	C,0	—	D,0
E	—	—	—	A,1

Basic notions

- For a completely specified m/c two states s_i and s_j of a machine M are compatible if and only if, for every input sequence the same output sequence will be produced regardless of whether s_i or s_j is the initial state
- However, for incompletely specified m/cs the next state many not be defined
- An input sequence is said to be applicable to a m/c if its state transitions can be definitely determined for a given input sequence (next state for each input symbol is specified)
- Behaviour of the m/cs may be observed when applicable sequences are applied when the m/c starts with either s_i or s_j as the initial state
- Outputs need not be specified for an applicable sequence; they need to match when specified

Compatible states
: Two states s_i and s_j of a 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 and regardless of whether s_i or s_j is the initial state

Compatibility issues

- Let s_i, s_j and s_k be states whose compatibilities are to be determined
- Let s_i^x and s_j^x be their x—successors on application of applicable sequence x
- What to do if the output of s_i^x is specified (as o_a) but that of s_j^x is not specified?
  - Since it is unspecified, it can be set to the specified output (o_a) so that s_i^x and s_j^x may be merged (consider merging of B and C of M_E)
- What to do if the output of s_k^x is specified (as o_b, $o_a \neq o_b$) but that of s_j^x is not specified, assuming that it is decided to merge s_i^x and s_j^x?
  - State s_j^x may be split; let s' be resultant of splitting s_j^x; now the output of s' can be set to the specified output (o_b) so that s_k^x and s' may be merged (consider merging of D and C of M_E)
- What to do if, on input a, the next state of s_i^x is specified (as s_a) but that of s_j^x is not specified?
  - Since it is unspecified, it can be set to the specified state (s_a) so that s_i^x and s_j^x may be merged
- What to do if, on input a, the next state of s_k^x is specified (as s_a', not compatibile with s_a) but that of s_j^x is not specified, assuming that it is decided to merge s_i^x and s_j^x?
  - State s_j^x may be split; let s' be resultant of splitting s_j^x; now the next state of s' can be set to the specified state (s_a') so that s_k^x and s' may be merged

Merger table

Table construction

- If s_i and s_j have output conflict on the next state, they are incompatible, indicated by a × in M_{ij}
  - States A and D have output conflict on I_1
  - States B and E have output conflict on I_1
- For each pair of states $\langle s_i, s_j \rangle$, M_{ij} indicates the required compatibilities of states
  - Compatibility of states A and B contingent of compatibility of E and F, also compatibility of A and B
  - Compatibility of states B and F contingent of compatibility of F and F, also compatibility of A and B
- Merger table is symmetric
  - Sufficient to have entries for s_i and s_j for $i > j$
- May be equivalently represented by a merger graph

FSM specification: M_E

PS	NS, output	
	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

B	EF, AB				
C	BC, EE	AC,EF			
D	×	×	EF, CD		
E	×	×	CE, CC	CD,CF	
F	DE, BB	~~AB,DF~~	BC,DE	~~BD, FD~~	BC,CD
	A	B	C	D	E

Table simplification

Unsatisfiable dependencies (direct)
: DF depending of compatibility of BD
: *Entries with (direct) unsatisfiable dependencies are crossed*

Unsatisfiable dependencies (transitive)
: AF depending of compatibility of DF
: *Entries with (transitive) unsatisfiable dependencies are crossed*

Self dependencies
: AB depending on AB, CD depending on CD, FD depending on FD
: *Self dependencies are dropped*

Reflexive dependencies
: Depending on compatibility of E with E, B with B
: *Reflexive dependencies are dropped*

Merger table after simplification

Merger table

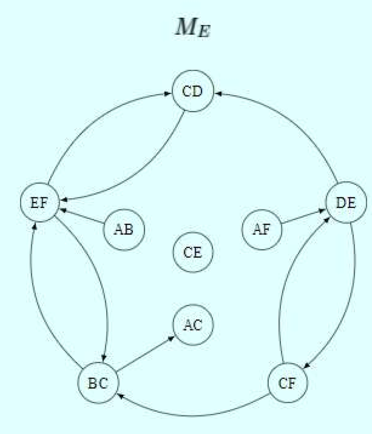
B	EF				
C	BC	AC,EF			
D	×	×	EF		
E	×	×	✓	CD,CF	
F	DE	×	BC,DE	×	BC,CD
	A	B	C	D	E

Compatibility graph

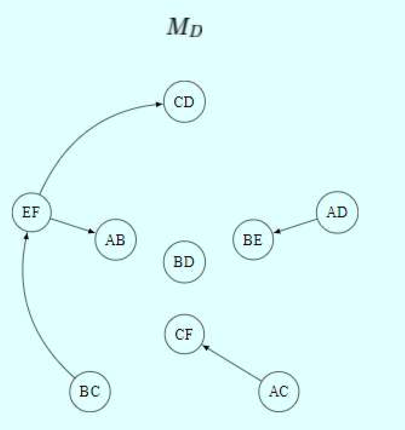
- Compatibility dependencies indicated in the merger table are depicted in the compatibility graph
- Each for each M_{ij} in the reduced merger table that is not crossed out, there is a node in compatibility graph
- If $\langle s_p, s_q \rangle \in M_{ij}$, there is a directed edge from the node for $\langle s_i, s_j \rangle$ to the node for $\langle s_p, s_q \rangle$, note that $\langle s_i, s_j \rangle \equiv \langle s_j, s_i \rangle$

Compatibility graphs

M_E

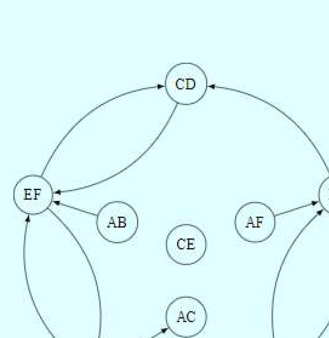


M_D



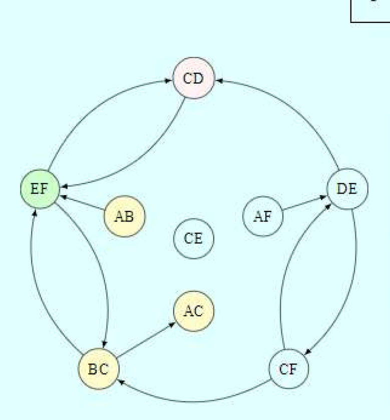
Closed set of compatibiles

- It is desirable to merge compatibile states
- For M_E, states C and E are compatibile and may be merged
- Similarly, states A and C are compatibile and may be merged
- However, states A and B are compatibile subject to E and F being compatibile; to merge A and B, E and F must also be merged
- States E and F are compatibile subject to compatibility of BC and CD; to merge EF, BC and CD must also be merged
- States B and C are compatibile subject to compatibility of AC; to merge BC, AC must also be merged
- Just CE may be merged without having to merge any other states, then there will be 5 states: A, B, CE, D, F so that all states are covered
- AB, EF, CD, BC and AC may be merged without having to merge any other states, then there will be states: AB, EF, CD, BC and AC, covering all states
  - However, AB, BC and AC together form a clique and may be merged to ABC
  - Then after, merging there will be only 3-states: ABC, CD and EF
  - Note that C had to be split between CD and ABC
- Thus, closed sets of compatibles need to be systematically (through branch and bound) explored and costed
- Cliques in a closed set of compatibiles may be combined to a single state
- Each closed set of compatible needs to be costed to retain the closed set of compatibles (after clique reduction) of least cosst
- The reduced FSM (NS and output function tables) need to be constructed



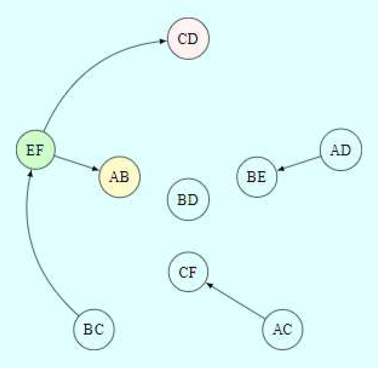
Costing of closed set of compatibiles for M_E

Sl	Closed set of compatibiles	Cost
1	A, B, CE, D, F	5
2	AB, EF, CD, BC	4
3	ABC, EF, CD	3
4	DE, CD, EF, BC, AC, CF	6
5	DE, CD, EF, ABC, CF	5
6	CDE, EF, ABC, CF	4



Reduced FSM for M_E

PS		NS, output	
Members	Symbol	I_1	I_2
ABC	α	δ,0	α,0
CD	γ	δ,1	γ,0
EF	δ	γ,1	α,0



Reduced FSM for M_D

PS		NS, output			
Members	Symbol	I_1	I_2	I_3	I_4
AB	α	γ,0	β,1	γ,1	α,1
CD	β	γ,0	γ,1	α,1	—
EF	γ	β,0	γ,0	α,0	β,1

Exercises

- Present the pseudocode for finding the minimum cost closed set of compatibles
- Work out the state minimisation for M_D and M_F