Checking states and transitions of a set of communicating finite state machines

R.M. Hierons

Professor of Computing in Brunel University

Yousra Lembachar

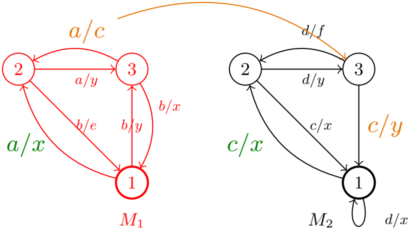
University of California Riverside

December 9, 2010

What is a model consisting of communicating finite state machines?

One FSM produces an output that is placed in the input queue of another FSM

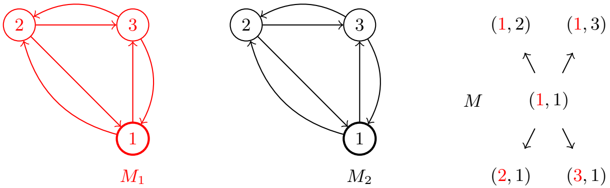
$M = M_1 | M_2$



- Global state (M) $= (s(M_1), s(M_2)), q(M_1), q(M_2))$
- A local transition is $(1, 2, a/x)$ and $(1, 2, c/x)$
- A global transition is ((3,3),(2,1),a/y)
- A stable state is when all the queues are empty
- (2,3) with b at the input queue of M_2 is not a stable state

Why don't we generate the product machine of these FSMs and apply standard methods?

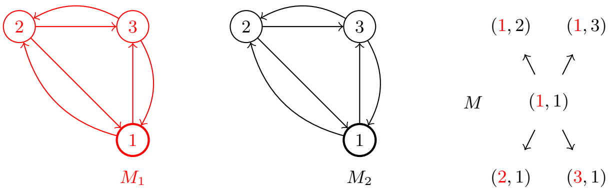
- If the model M has n CFSMs, each CFSM i having n_i states,
  - The number of the transitions of M is $O(|X|\Pi_{i=1}^{i=n}(ni)$



The potential states of M are ((1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3))

Why don't we generate the product machine of these FSMs and apply standard methods?

- If the model M has n CFSMs, each CFSM i having n_i states,
  - The number of the transitions of M is $O(|X|\Pi_{i=1}^{i=n}(ni)$



Checking only local transitions $\Rightarrow O(\Sigma_{i=1}^{i=n}|X_i|n_i)$

Outline

Assumptions

Avoiding fault masking while testing local and global transitions

Checking local states

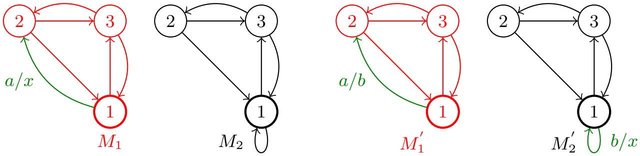
Checking global states

Assumptions

- $M = M_1|...|M_n$
- No errors in communications and queueing
  - Local transitions correct $\Rightarrow$ Global transitions correct
- M_i has one initial state
- M_i is deterministic, minimal, strongly connected and completely specified
  - The input alphabets of the M_i are disjoint
- M is a deterministic model, deadlock and live-lock free
- Only stable states are considered
- M is equivalent to the product machine
- Only output errors and transfer errors are considered

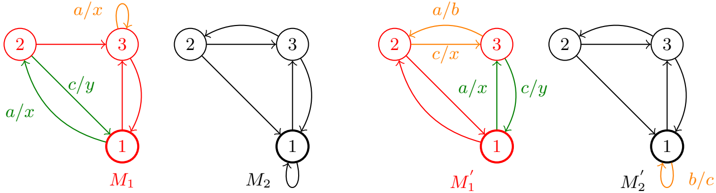
Fault masking

- Masking an output fault



$$(\mathbf{1},1),(\mathbf{2},1),a/x)$$

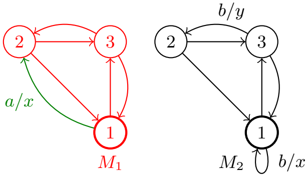
- Masking a state transfer fault



$$((1,1),(1,1),ac/xy)\ ((3,1),(3,1),a/x)$$

Avoiding fault masking

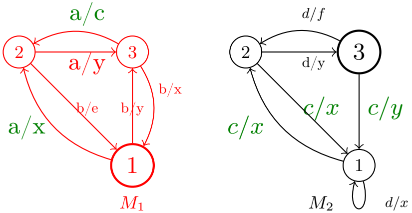
- Assumption: When testing a local transition t, all other transitions executed are correct
  - Finding a set of global transitions that contain t that allow any fault in t to be revealed



- A test from (1,1) with a will not reveal the fault since the output $= x$
- A test from $\boxed{(1,3)}$ with a will reveal the fault since the output $= y$

Checking local states

- Finding the input sequence u that may check s **for some set of states of the other** $M_j \in M$



- $(1, \_)\ a/x$
- $(3, \_)\ a/c$
- $(1/2, \_)\ c/x$
- $(3, \_)\ c/y$

a checks that M_1 in state 1 iff M_2 is in state 3.

$\Rightarrow$ Constrained identification sequence CIS

Checking global states

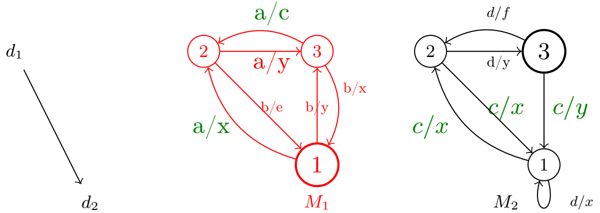
- Choose a CIS for each local state and execute the test sequence
  ... but, there are maybe some dependencies in the CIS!

$$\left.\begin{array}{l}\text{Checking } s_i \Rightarrow M_j \text{ in } s_j \text{ and } s_j \text{ correct}\\ \text{Checking } s_j \Rightarrow M_i \text{ in } s_i \text{ and } s_i \text{ correct}\end{array}\right\}\text{if } s_i \text{ and } s_j \text{ are incorrect?}$$

$\Rightarrow$ Dependency circularity

Dependency digraph

Directed graph $G_D = (V_D, E_D)$ where V_D is $(d_1, ..., d_n)$ and d_i representes M_i.

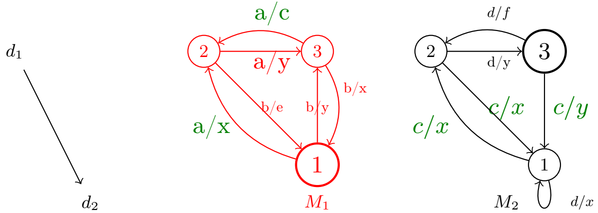


$$\left.\begin{array}{r} \text{CIS}_1\text{: We can use } a \text{ to check state } {\color{red}1} \\ \text{iff } M_2 \text{ is in state } 3 \\ \text{CIS}_2\text{: We can use } c \text{ to check } 3 \end{array}\right\} \Rightarrow \text{Cycle free graph}$$

$\Rightarrow$ We can use these CIS to test the final global state (1,3).

Dependency digraph

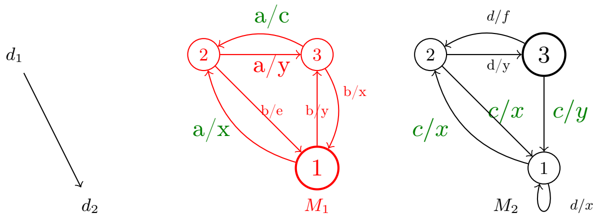
Directed graph $G_D = (V_D, E_D)$ where V_D is $(d_1, ..., d_n)$ and d_i representes M_i.



$$\left.\begin{array}{r l} \text{CIS}_1\text{:} & \text{We can use } a \text{ to check state } {\color{red}1} \\ & \text{iff } M_2 \text{ is in state } 3 \\ \text{CIS}_2\text{:} & \text{We can use } c \text{ to check } 3 \end{array}\right\} \Rightarrow \text{Cycle free graph}$$

$$(c/x, d/y, c/y), \text{ reset, } (c/x, d/y, a/x)$$

Sequencing CIS

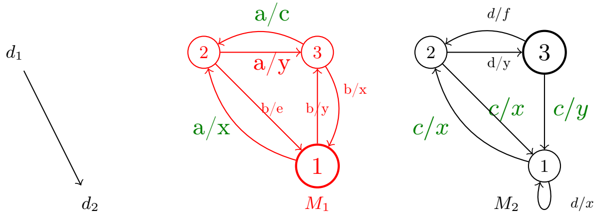


- The edges of the dependency graph impose an ordering that may reduce the test effort.

$$\begin{tikzcd} d_1 \arrow[dr] & d_3 \arrow[l] \\ d_3 & d_2 \arrow[l] \arrow[u] \end{tikzcd} \qquad \begin{tikzcd} O_1 \arrow[dr] & O_3 \arrow[l] \\ O_4 & O_2 \arrow[l] \arrow[u] \end{tikzcd}$$

These CISs cannot be sequenced since there is a cycle.
Partitioning the set of CIS $\Rightarrow$ many cycle free order digraphs.

Sequencing CIS

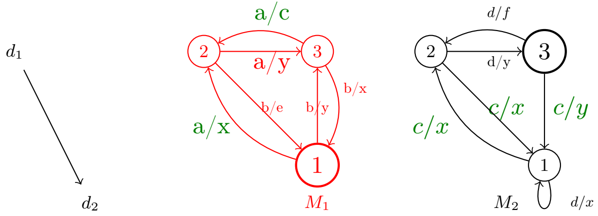


- Edge from d_1 to d_2 $\Rightarrow$ u_1 depends on $s(M_2)$ $\Rightarrow$ u_1 before u_2 since (u_2 will change $s(M_2)$.)

$$\begin{tikzcd}
d_1 \arrow[dr] & d_3 \arrow[l] \\
d_3 & d_2 \arrow[l] \arrow[u]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
O_1 \arrow[dr] & O_3 \arrow[l] \\
O_4 & O_2 \arrow[l] \arrow[u]
\end{tikzcd}$$

These CISs cannot be sequenced since there is a cycle.
Partitioning the set of CIS $\Rightarrow$ many cycle free order digraphs.

Sequencing CIS



$(c/x, d/y, a/x, c/y)$ instead of $(c/x, d/y, c/y)$, reset, $(c/x, d/y, a/x)$

$$\begin{tikzcd} d_1 \arrow[dr] & d_3 \arrow[l] \\ d_3 & d_2 \arrow[l] \arrow[u] \end{tikzcd} \qquad \begin{tikzcd} O_1 \arrow[dr] & O_3 \arrow[l] \\ O_4 & O_2 \arrow[l] \arrow[u] \end{tikzcd}$$

These CISs cannot be sequenced since there is a cycle.
Partitioning the set of CIS $\Rightarrow$ many cycle free order digraphs.

Conclusions

- An interesting approach when testing a model consisting of CFSMS.
- Testing transitions and checking states using constrained identification sets
  $\Rightarrow$ avoids generating the product machine.
- CIS $\Rightarrow$ circuit of dependencies
  $\Rightarrow$ finding a consistent set of CIS with a circuit free digraph.
  $+$ sequencing is possible to reduce the test effort.
- No focus on how to generate the CIS or how to get a circuit free order digraph.