Introduction to Software Testing
Chapter 8.5
Logic Coverage for FSMs

Paul Ammann & Jeff Offutt

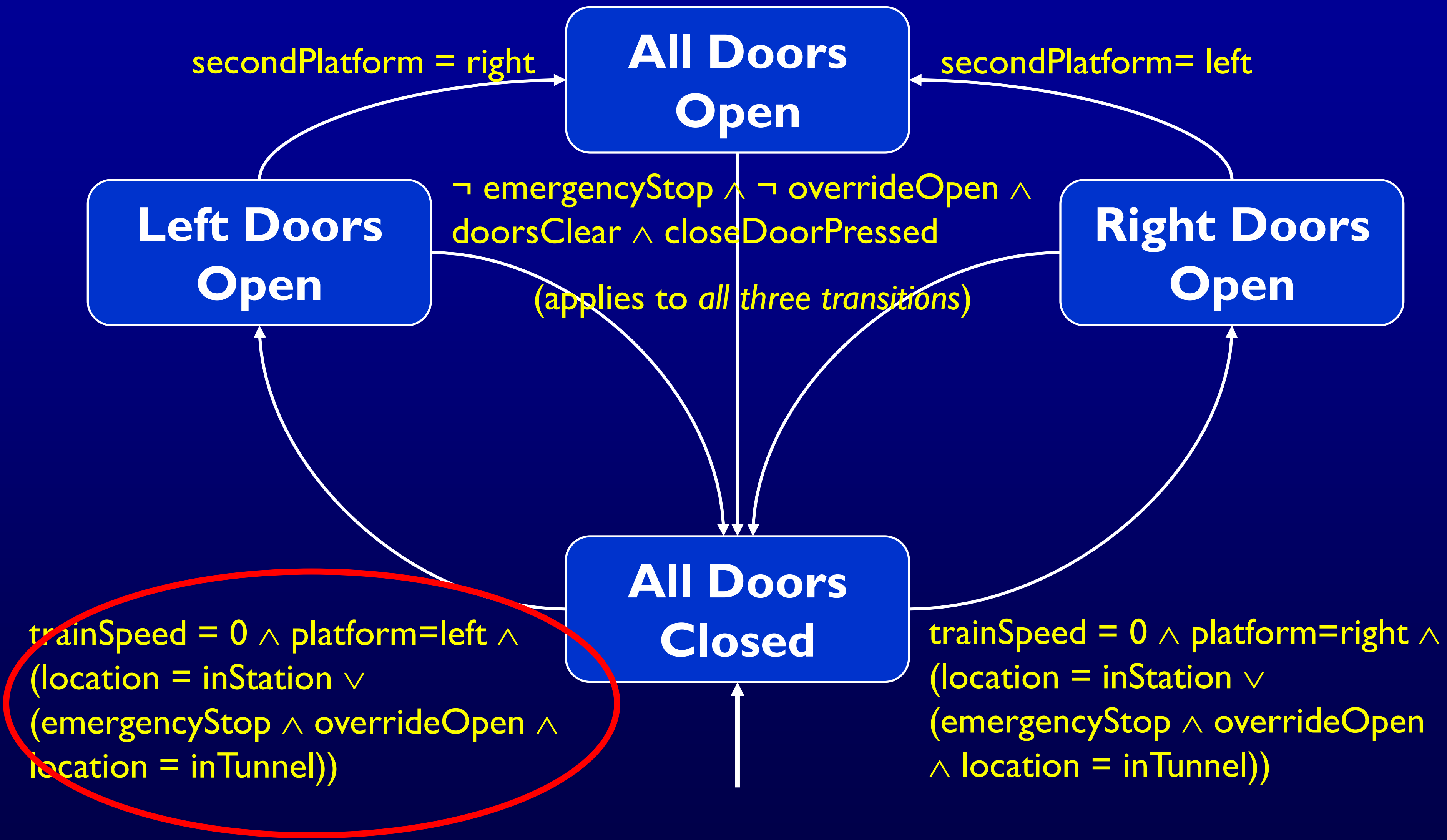
http://www.cs.gmu.edu/~offutt/softwaretest/

Covering Finite State Machines

- FSMs are graphs
  - Nodes represent state
  - Edges represent transitions among states
- Transitions often have logical expressions as guards or triggers
- As we said :

Find a *logical expression* and cover it

Example—Subway Train



All Doors Open

Left Doors Open

Right Doors Open

All Doors Closed

secondPlatform = right

secondPlatform= left

¬ emergencyStop ∧ ¬ overrideOpen ∧ doorsClear ∧ closeDoorPressed

(applies to *all three transitions*)

trainSpeed = 0 ∧ platform=left ∧ (location = inStation ∨ (emergencyStop ∧ overrideOpen ∧ location = inTunnel))

trainSpeed = 0 ∧ platform=right ∧ (location = inStation ∨ (emergencyStop ∧ overrideOpen ∧ location = inTunnel))

Determination of the Predicate

trainSpeed = 0 ∧ platform=left ∧ (location = inStation ∨ (emergencyStop ∧ overrideOpen ∧ location = inTunnel))

Find the truth assignments that let the all six clauses determine the value of the predicate.

That is, solve for $P_{trainSpeed}$, then $P_{platform=left}$, etc.

$a \wedge b \wedge (c \vee (d \wedge e \wedge f))$

Determination of the Predicate

$\text{trainSpeed} = 0 \wedge \text{platform}=\text{left} \wedge (\text{location} = \text{inStation} \vee (\text{emergencyStop} \wedge \text{overrideOpen} \wedge \text{location} = \text{inTunnel}))$

$P_{\text{trainSpeed} = 0}$: $\text{platform} = \text{left} \wedge (\text{location} = \text{inStation} \vee (\text{emergencyStop} \wedge \text{overrideOpen} \wedge \text{location} = \text{inTunnel}))$

Solution for $P_{\text{trainSpeed}}$ …

$P_{\text{platform} = \text{left}}$: $\text{trainSpeed} = 0 \wedge (\text{location} = \text{inStation} \vee (\text{emergencyStop} \wedge \text{overrideOpen} \wedge \text{location} = \text{inTunnel}))$

Solution for P_{platform} …

$P_{\text{location} = \text{inStation}}$: $\text{trainSpeed} = 0 \wedge \text{platform} = \text{left} \wedge (\neg\, \text{emergencyStop} \vee \neg\, \text{overrideOpen} \vee \neg\, \text{location} = \text{inTunnel})$

Solution for $P_{\text{inStation}}$ …

$P_{\text{emergencyStop}}$: $\text{trainSpeed} = 0 \wedge \text{platform} = \text{left} \wedge (\neg\, \text{location} = \text{inStation} \wedge \text{overrideOpen} \wedge \text{location} = \text{inTunnel})$

Solution for $P_{\text{emergencyStop}}$ …

$P_{\text{overrideOpen}}$: $\text{trainSpeed} = 0 \wedge \text{platform} = \text{left} \wedge (\neg\, \text{location} = \text{inStation} \wedge \text{emergencyStop} \wedge \text{location} = \text{inTunnel})$

Solution for $P_{\text{overrideOpen}}$ …

$P_{\text{location} = \text{inTunnel}}$: $\text{trainSpeed} = 0 \wedge \text{platform} = \text{left} \wedge (\neg\, \text{location} = \text{inStation} \wedge \text{emergencyStop} \wedge \text{overrideOpen})$

Solution for P_{location} …

Test Truth Assignments (CACC)

trainSpeed = 0 ∧ platform=left ∧ (location = inStation ∨ (emergencyStop ∧ overrideOpen ∧ location = inTunnel))

Major Clause	Speed=0	platform=left	inStation	emergStop	overrideOpen	inTunnel
trainSpeed = 0	**T**	t	t	t	t	t
trainSpeed != 0	**F**	t	t	t	t	t
platform = left						
platform != left						
inStation						
¬ inStation						
emergencyStop						
¬ emergStop						
overrideOpen						
¬ overrideOpen						
inTunnel						
¬ inTunnel						

One of these must be true

Fill in the remaining truth assignments based on the expressions computed for the previous slide

Test Truth Assignments (CACC)

$\text{trainSpeed} = 0 \wedge \text{platform}=\text{left} \wedge (\text{location} = \text{inStation} \vee (\text{emergencyStop} \wedge \text{overrideOpen} \wedge \text{location} = \text{inTunnel}))$

Major Clause	Speed=0	platform=left	inStation	emergStop	overrideOpen	inTunnel
trainSpeed = 0	**T**	t	t	t	t	t
trainSpeed != 0	**F**	t	t	t	t	t
platform = left	t	**T**	t	t	t	t
platform != left	t	**F**	t	t	t	t
inStation	t	t	**T**	f	f	f
¬ inStation	t	t	**F**	f	f	f
emergencyStop	t	t	f	**T**	t	t
¬ emergStop	t	t	f	**F**	t	t
overrideOpen	t	t	f	t	**T**	t
¬ overrideOpen	t	t	f	t	**F**	t
inTunnel	t	t	f	t	t	**T**
¬ inTunnel	t	t	f	t	t	**F**

One of these must be true

One of these must be false

Problem With a Predicate?

	trainSpeed=0	platform=left	inStation	emergencyStop	overrideOpen	inTunnel
inStation	t	t	**T**	f	f	f
¬ inStation	t	t	**F**	f	f	f

Think about these two values …

The model only has two locations

inStation and *inTunnel*

So these cannot both be false!

If the train is not in the station (*location != inStation*), then it must be in a tunnel (*location = inTunnel*)

Possible solutions :

1. Check with the developer for mistakes (do this first)
2. Rewrite the predicate to eliminate dependencies (if possible)
3. Change truth assignment : t t F f f t

Expected Results

Expected outputs are read from the FSM :

- When the major clause is true, the transition is taken
- When false, the transition is not taken

	Expected Results
trainSpeed = 0 trainSpeed != 0	
platform = left platform != left	
inStation ¬ inStation	
emergencyStop ¬ emergencyStop	
overrideOpen ¬ overrideOpen	
inTunnel ¬ inTunnel	

Fill in the expected results

Expected Results

Expected outputs are read from the FSM :

- When the major clause is true, the transition is taken
- When false, the transition is not taken

	Expected Results
trainSpeed = 0	Left Doors Open
trainSpeed != 0	All Doors Closed
platform = left	Left Doors Open
platform != left	All Doors Closed
inStation	Left Doors Open
¬ inStation	All Doors Closed
emergencyStop	Left Doors Open
¬ emergencyStop	All Doors Closed
overrideOpen	Left Doors Open
¬ overrideOpen	All Doors Closed
inTunnel	Left Doors Open
¬ inTunnel	All Doors Closed

Do you notice anything “funny”?

If *platform !=left*, then *platform* must equal ***right***

So the expected output of this test is to go to state “**Right Doors Open**”

Accidental transitions must be recognized when designing expected results during test automation

Early Identification is a Win!

The process of modeling software artifacts for test design can help us find defects in the artifacts

This is a very powerful side-effect of the model-driven test design process

Complicating Issues

- Some buttons must be pressed simultaneously to have effect – so timing must be tested
- Reachability : The tests must reach the state where the transition starts (the prefix)
- Exit : Some tests must continue executing to an end state
- Expected output : The expected output is the state that the transition reaches for true values, or same state for false values
- Accidental transitions : Sometimes a false value for one transition happens to be a true value for another
  - The alternate expected output must be recognized

Test Automation Issues

- Mapping problem : The names used in the FSMs may not match the names in the program
- Examples
  - *platform = left* requires the train to go to a specific station
  - *trainspeed = 0* probably requires the brake to be applied multiple times
- The solution to this is implementation-specific
  - Sometimes a direct name-to-name mapping can be found
  - Sometimes more complicated actions must be taken to assign the appropriate values
  - Simulation : Directly inserting value assignments into the middle of the program
- This is an issue of controllability

Summary FSM Logic Testing

- FSMs are widely used at all levels of abstraction
- Many ways to express FSMs
  - Statecharts, tables, Z, decision tables, Petri nets, …
- Predicates are usually explicitly included on the transitions
  - Guards
  - Actions
  - Often represent safety constraints
- FSMs are often used in embedded software