

Introduction to Software Testing

Chapter 3.5

Logic Coverage for FSMs

Paul Ammann & Jeff Offutt

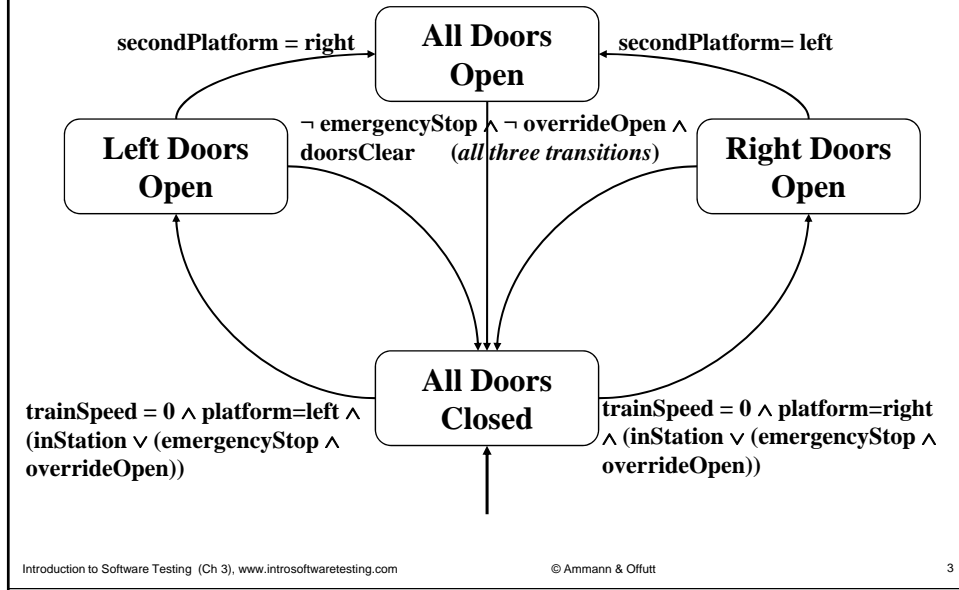
www.introsoftwaretesting.com

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



Determination of the Predicate

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

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

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

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

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

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

Test Truth Assignments (CACC)

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

	trainSpeed=0	platform=left	inStation	emergencyStop	overrideOpen
trainSpeed = 0	T	t	t	t	t
trainSpeed != 0	F	t	t	t	t
platform = left	duplicate	T	t	t	t
platform != left	t	F	t	t	t
inStation	t	t	T	f	f
\neg inStation	t	t	F	f	f
emergencyStop	t	t	f	T	t
\neg emergencyStop	t	t	f	F	t
overrideOpen	duplicate	t	f	t	T
\neg overrideOpen	t	t	f	t	F

Introduction to Software Testing (Ch 3), www.introssoftwaretesting.com

© Amr

Note : other choices are possible

5

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

Introduction to Software Testing (Ch 3), www.introssoftwaretesting.com

© Ammann & Offutt

6

Test Scripts

- **Test scripts** are executable sequences of value assignments
- **Mapping problem** : The names used in the FSMs may not match the names in the program
 - 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
- The solution to this is **implementation-specific**

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