CS342: Software Engineering

Dr. Ismail Hababeh
German Jordanian University

Lecture 11
ANALYSIS OF DYNAMIC SYSTEMS

Adapted from Software Engineering, by Dr. Paul E. Young
& slides by Dr. Mohammad Daoud

Analysis of a Finite-State Machine

- A Finite State Machine FSM is a mathematical model of computation based on a hypothetical machine made of one or more states where particular inputs cause particular changes in state.
- FSM model is built by determining the relationships among the inputs, the outputs, and the states of the flip-flops (circuit that has two stable states and can be used to store state information).

Analysis of a Finite-State Machine

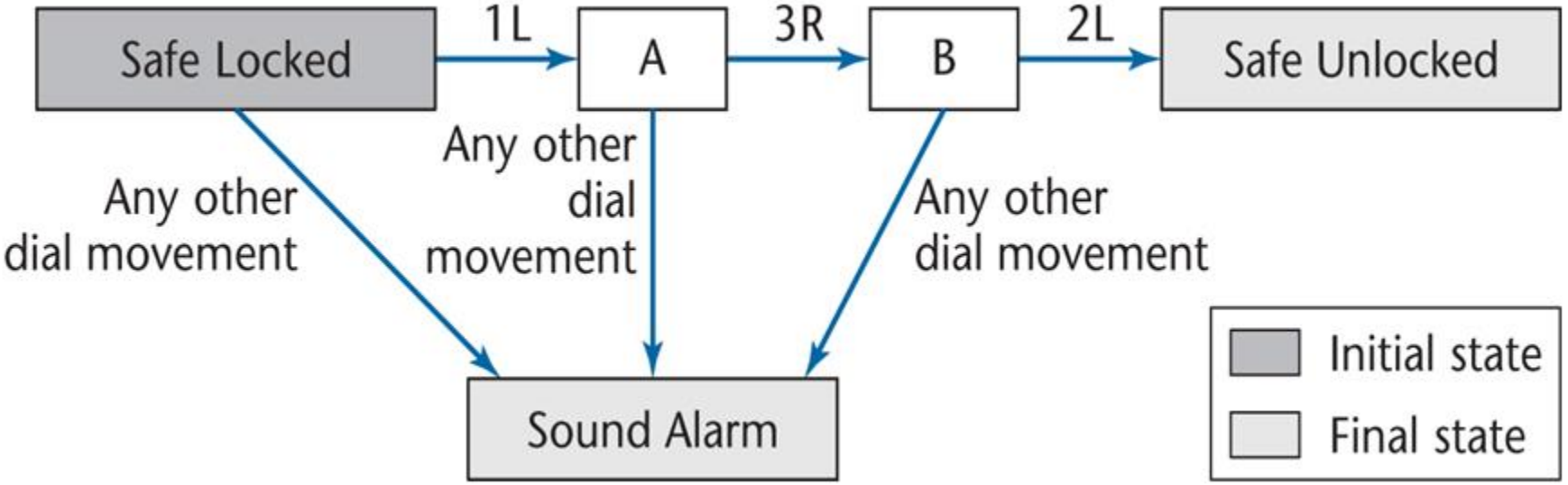
- The machine has a transition function from one state to another to perform different actions.
- Only one single state of this machine can be active at the same time.

Analysis of a Finite-State Machine

- FSM is used to design both computer programs and sequential logic circuits.
- It is designed as an abstract *machine* that can be in one of a *finite* number of *states*.

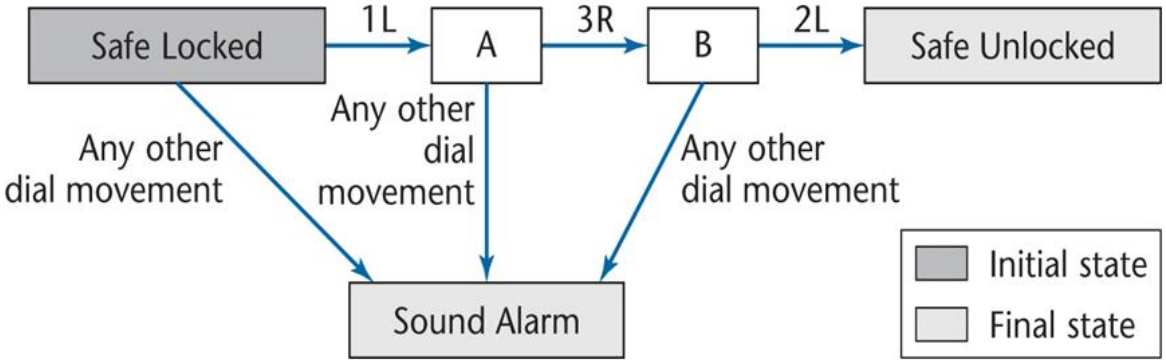
Finite State Machines – Locker Case Study

A security system has a combination lock that can be in one of three positions, labeled 1, 2, and 3. The dial can be turned left or right (L or R). Thus, there are six possible dial movements, namely 1L, 1R, 2L, 2R, 3L, and 3R. The combination to the safe unlock is 1L, 3R, 2L; any other dial movements will turn on the alarm.



Analysis of the Lock Case Study

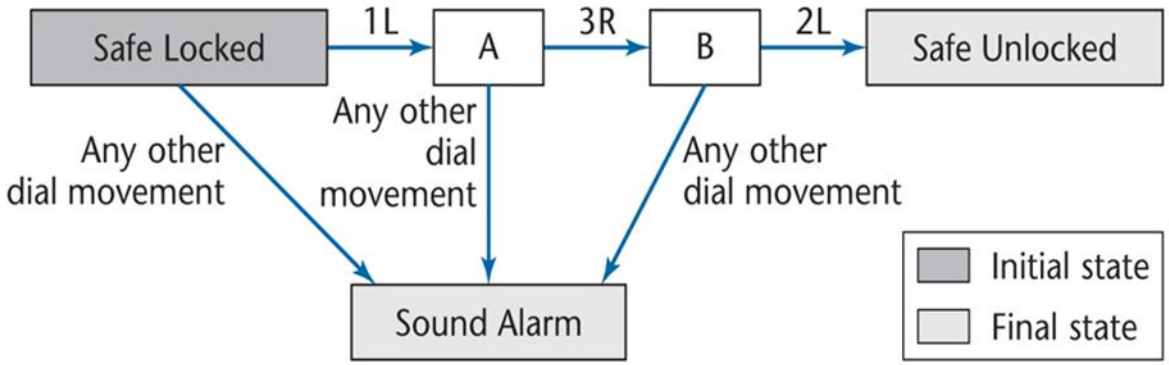
- The set of states J is {Safe Locked, A, B, Safe Unlocked, Sound Alarm}
- The set of inputs K is {1L, 1R, 2L, 2R, 3L, 3R}
- The initial state of J is Safe Locked
- The set of final states of J is {Safe Unlocked, Sound Alarm}
- The transition function T (*next slide*)



Finite State Machines – Transition Function

Transition table

Dial Movement \ Current State	Table of Next States Safe Locked	A	B
1L	A	Sound alarm	Sound alarm
1R	Sound alarm	Sound alarm	Sound alarm
2L	Sound alarm	Sound alarm	Safe unlocked
2R	Sound alarm	Sound alarm	Sound alarm
3L	Sound alarm	Sound alarm	Sound alarm
3R	Sound alarm	B	Sound alarm



Finite State Machines - Elevator Case Study

A product is to be installed to control **N** elevators in a building with **M** floors. The problem concerns the logic required to move elevators between floors according to the set of constraints.

Analysis of the Elevator Control Logic

1. Each elevator has a set of ***m*** buttons, one for each floor.

– These buttons are lightened on when pressed and cause the elevator to visit the corresponding floor.

– The button is lightened off when the corresponding floor is visited by the elevator.

Analysis of the Elevator Control Logic

2. Each floor, except the first and the top floors, has two buttons, one to request an elevator Up, one to request an elevator Down.
   - These buttons are lightened on when pressed.
   - The light is off when an elevator visits the floor, then moves in the desired direction
3. If an elevator has no requests
   - It remains at its current floor.
   - The door closed.

Analysis of the Elevator Control Logic

- There are two sets of buttons
  - Elevator buttons

    In each elevator, one for each floor

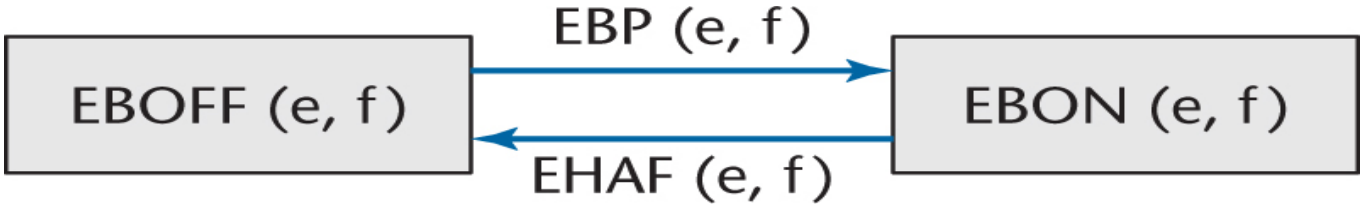
    Elevator Button: states, events, predicates, and transitions.
  - Floor buttons

    Two on each floor, one for elevator up, one for elevator down.

    Floor Button: states, events, predicates, and transitions.

Elevator Buttons (EB) States

- **EB (e, f):** Elevator Button in elevator e pressed to request floor f
- **Two States**
  - EBON (e, f): Elevator Button (e, f) ON
  - EBOFF (e, f): Elevator Button (e, f) OFF



  - If an elevator button is on and the elevator arrives at floor f, then the elevator button is turned off
  - If the elevator button is off and the elevator button is pressed, then the elevator button becomes on

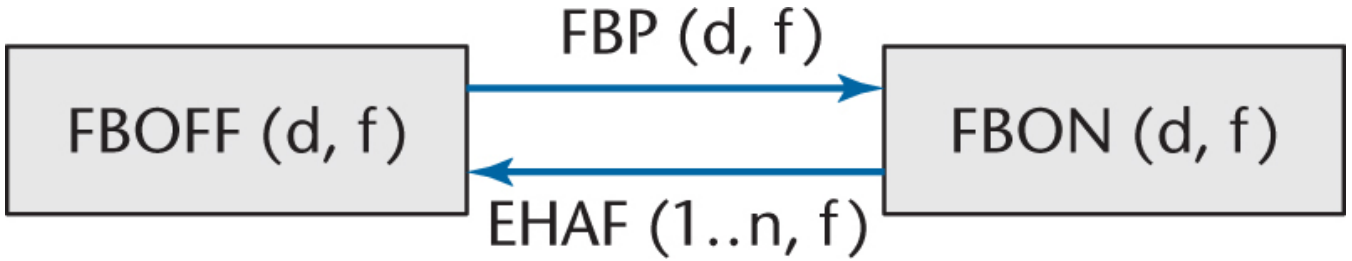
Elevator Buttons – Events - Predicates - Transitions

- Events:
  - EBP (e, f): Elevator Button (e, f) Pressed
  - EHAF (e, f): Elevator e Has Arrived at Floor f
- Predicates:

  V (e, f): Elevator e is Visiting (stopped at) floor f
- Transitions: Transition Rules
  1. EBOFF (e, f) and EBP (e, f) and not V (e, f) $\Rightarrow$ EBON (e, f)
  2. EBON (e, f) and EHAF (e, f) $\Rightarrow$ EBOFF (e, f)

Floor Buttons (FB) States

- FB (d, f):
  - Floor Button on floor f that requests elevator traveling in direction d
- Two states
  - FBON (d, f): Floor Button (d, f) ON
  - FBOFF (d, f): Floor Button (d, f) OFF



  - If the floor button is on and an elevator arrives at floor f, traveling in the correct direction d, then the floor button is turned off
  - If the floor button is off and the floor button is pressed, then the floor button is turned on

Floor Buttons – Events - Predicates -Transitions

- Events:
  - FBP (d, f): Floor Button (d, f) Pressed
  - EHAF (1..n, f): Elevator 1 or … n Has Arrived at Floor f
- Predicates:

  S (d, e, f): Elevator e is visiting floor f

  Direction of motion is up (d = U), down (d = D), or no requests are pending (d = N)
- Transitions: Transition Rules
  - FBOFF (d, f) and FBP (d, f) and not S (d, 1..n, f) ⇒ FBON (d, f)
  - FBON (d, f) and EHAF (1..n, f) and S (d, 1..n, f) ⇒ FBOFF (d, f), d = U or D

Elevator Component Sub-States

- The elevator states consists of component sub-states, including:
  - Elevator move
  - Elevator wait
  - Elevator stop
  - Door open with timer running
  - Door close after a timeout
- We assume that the elevator controller directs the elevator through the sub-states

Elevator Component Sub-States

- Elevator sub-states
  - M (d, e, f): Move in direction d (floor f is next)
  - W (e, f): Wait at floor f (door closed)
  - S (d, e, f): Stop (d-bound) at floor f
    - For simplicity, 3 stopped states are considered
      - S (U, e, f)
      - S (N, e, f)
      - S (D, e, f)

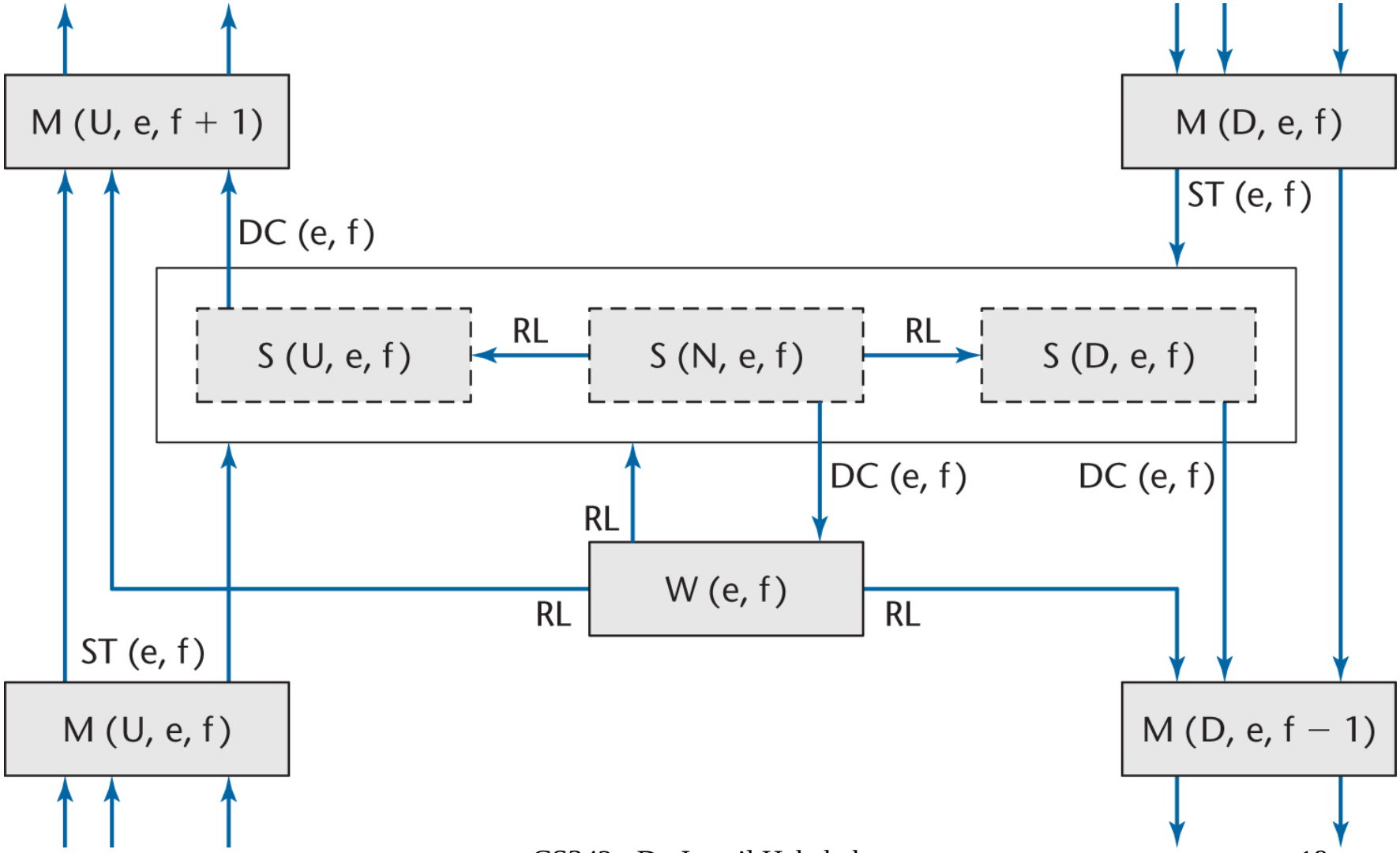
  and generalized in one state S (d, e, f).

Elevator Events and Transitions

- Elevator Events
  - DC (e, f): Door Closed of elevator e, floor f
  - ST (e, f): Sensor Triggered as elevator e near floor f
  - RL: Request Logged (button pressed)
- Elevator Transitions

  If the elevator e is in state S (d, e, f), door closed, button is pressed, then elevator e will move up, down, or go into wait state
  - DC (e, f)+ S (U, e, f)+ and Button pressed up ⇒ M (U, e, f+1)
  - DC (e, f)+ S (D, e, f)+ and Button pressed down ⇒ M (D, e, f-1)
  - DC (e, f)+ S (N, e, f)+ and no Button pressed ⇒ W (e, f)

Elevator State Transition Diagram



The Power of Finite State Machines (FSM)

- Specifies complex systems
- There is no need for complex pre-conditions and post-conditions.
- Next state take the simple form

current state + event + predicate $\Rightarrow$ next state

The Power of Finite State Machines (FSM)

- Advantages of using an FSM, a specification is
  - Easy to write it down
  - Easy to validate
  - Easy to convert into a design
  - Easy to convert into code
  - Easy to understand
  - Easy to maintain
  - More precise than graphical methods
- Disadvantages of FSM
  - Timing considerations are not handled

Difficulty with Specifying Real-Time Systems

- A major difficulty with specifying real-time systems is timing
  - Synchronization problems
  - Race conditions (*perform two or more operations at the same time*)
  - Deadlock
- Often a consequence of poor specifications
- Solution? ⇒ Petri Nets.