Computational Models for Embedded Systems

Vescan Andreea, PHD, Assoc. Prof.

Faculty of Mathematics and Computer Science
Babeș-Bolyai University
Cluj-Napoca

2025-2026

Lecture 7a: Finite State Machines (1)

Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

(Benjamin Franklin)

Outline

- FSM – definition + example
- FSM – advantages & disadvantages
- FSM – modeling concurrency ➔ composition
- Moore vs. Mealy

- **FSM – Demo - Traffic-Pedestrian**
  - FSMs – Traffic, Pedestrian, Traffic – Pedestrian – Composition
  - Traffic-Pedestrian – Electronic Circuit
  - Traffic-Pedestrian – Nucleo – project (video)
  - Traffic-Pedestrian – Model checking - JSpin
- Next lecture:
  - **Invited lecture**
  - **Accenture**

FSM – definition

- FSM = (I, O, S, r, δ, λ)
  - I = input
  - O = output
  - S = states
  - r=final states?
  - δ : IxS→S – state transition function
  - λ: IxS→O – output function

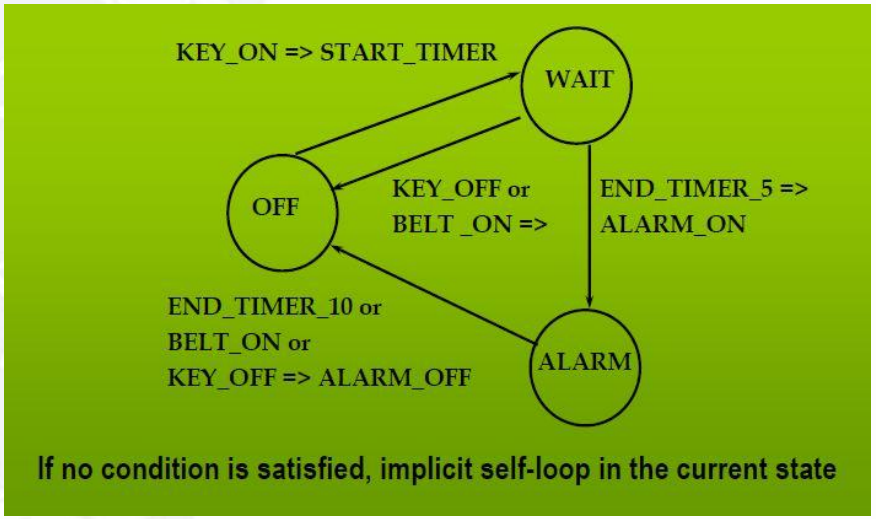
FSM – example

- If the driver

  turns on the key, and

  does not fasten the seat belt within 5 seconds
- then an alarm beeps for 5 seconds,
- or
- until the driver fastens the seat belt, or

  until the driver turns off the key



FSM - example

- FSM = (I, O, S, r, δ, λ)
- I={KEY_ON, KEY_OFF,BELT_ON,END_TIMER_5, END_TIMER_10}
- O={START_TIMER, ALARM_ON, ALARM_OFF}
- S={OFF, WAIT, ALARM}
- r=OFF

Set of all subsets of I (implicit "and")

All other inputs are implicitly absent

- δ :2'xS→S

  e.g. ({KEY_OFF}, WAIT)=OFF

- λ :2'xS-2^{O}

  e.g. λ({KEY_ON}, OFF}={START_TIMER}

FSM

Advantages	Disadvantages

• Easy to use (graphical languages)
• Powerful algorithms for
  • synthesis (SW and HW)
  • verification
	• Sometimes over-specify implementation (sequencing is fully specified)
	• Number of states can be unmanageable
	• Numerical computations cannot be specified compactly (need Extended FSMs)

Modeling Concurrency

- Need to compose parts described by FSMs
- Describe the system using a number of FSMs and interconnect them
- How do the interconnected FSMs talk to each other?

- Given (I-input, O-output, S-states, δ :IxS→S state transition function, λ:IxS→O output function)
  - $M_1=(I_1, O_1, S_1, r_1, \delta_1, \lambda_1)$ and
  - $M_2=(I_2, O_2, S_2, r_2, \delta_2, \lambda_2)$
  - Find the composition
    - $M=(I, O, S, r, \delta, \lambda)$
    - Given a set of constraints of the form:
    - $C=\{(o, i_1, ..., i_n)$: o is connected to $i_1, ..., i_n\}$

FSM Composition

- **Bridle complexity via hierarchy:**

  FSM product yields an FSM

- **Fundamental hypothesis:**
  - all the FSMs change state together (synchronicity)

- **System state** = Cartesian product of component states
  - (state explosion may be a problem...)

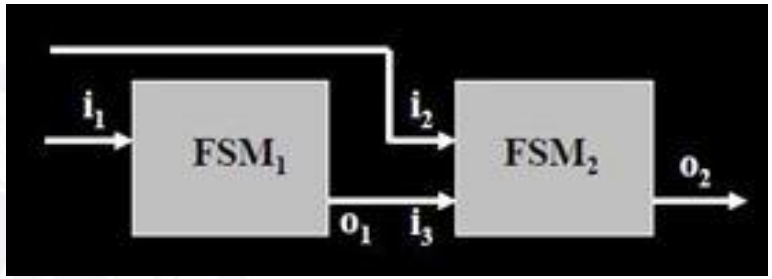
FSM Composition

- Unconditional product M'=(I', O', S', r', δ', λ')
- I'= $I_1 \cup I_2$
- O'= $O_1 \cup O_2$
- S'=$S_1 \times S_2$
- r'=$r_1 \times r_2$
- δ'=$\{(A_1,A_2,s_1,s_2,t_1,t_2): (A_1,s_1,t_1) \in \delta_1$ and $(A_2,s_2,t_2) \in \delta_2\}$
- λ '=$\{(A_1,A_2,s_1,s_2,B_1,B_2): (A_1,s_1,B_1) \in \delta_1$ and $(A_2,s_2,B_2) \in \delta_2\}$
- Note:
  - $A_1 \subseteq I_1$, $A_2 \subseteq I_2$, $B_1 \subseteq O_1$, $B_2 \subseteq O_2$
  - $2^{X \cup Y}=2^X \times 2^Y$
- Constraint application
  - $\lambda=\{(A_1, A_2, s_1,s_2,B_1,B_2) \in \lambda'$:for all $(o,i_1, \ldots, i_n) \in C$ o $\in B_1 \cup B_2$ if and only if $i_j \in A_1 \cup A_2$ for all j}
- The application of the constraint rules out the cases where the connected input and output have different values (present/absent).

FSM Composition

- $I = I_1 \cup I_2$
- $O = O_1 \cup O_2$
- $S = S_1 \times S_2$

$$o_1 \in O_1, i_3 \in I_2, o_1 = i_3 \ (communication)$$



- Assume that

$$\delta_1(\{i_1\}, s_1) = t_1, \lambda_1(\{i_1\}, s_1) = \{o_1\}$$

$$\delta_2(\{i_2, i_3\}, s_2) = t_2, \lambda_2(\{i_2, i_3\}, s_2) = \{o_2\}$$

- We have:

$$\delta(\{i_1, i_2, i_3\}, (s_1, s_2) = (t_1, t_2)$$

$$\lambda(\{i_1, i_2, i_3\}, (s_1, s_2)) = \{o_1, o_2\}$$

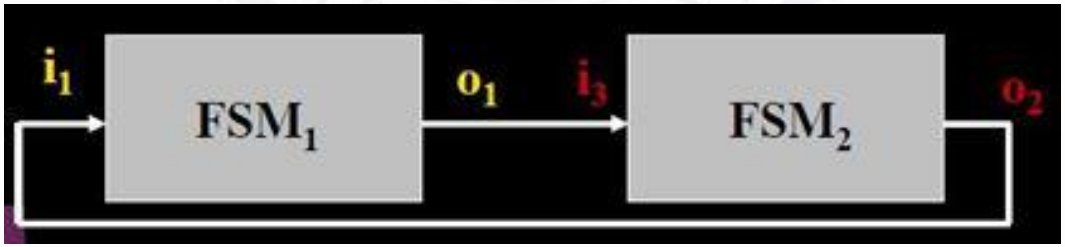
- i.e. i_3 is in input pattern iff o_2 is in output pattern.

FSM Composition

- Problem: what if there is a cycle?
  - Moore machine: δ depends on input and state, λ only on state
    - Composition is always *well-defined*
  - Mealy machine: δ and λ depend on input and state
    - Composition may be *undefined*

$$What\ if\ \lambda_1(\{i_1\}, s_1) = \{o_1\}\ but\ o_2 \notin \lambda_2(\{i_3\}, s_2)?$$

- Causality analysis in Mealy FSMs (Berry '98)



Moore vs. Mealy

- Theoretically, same computational power (almost)
- In practice, different characteristics.

Moore machines

- non-reactive
  (response delayed by 1 cycle)
- easy to compose
  (always well-defined)
- good for implementation
- software is always “slow”
- hardware is better when I/O is latched

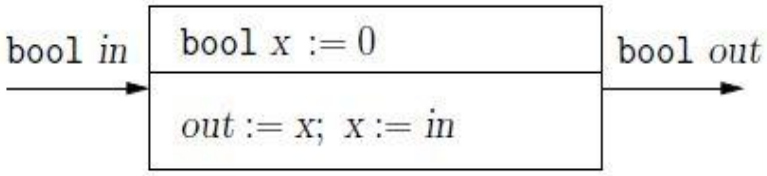
Mealy machines

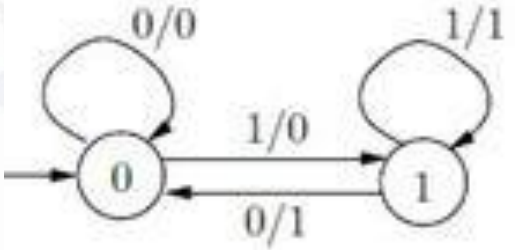
- reactive (0 response time)
- hard to compose (problem with combinational cycles)
- problematic for implementation
- software must be “fast enough”
  (synchronous hypothesis)
- may be needed in hardware, for speed

FSM – Mealy

- Finite-state components →behavior → a labeled finite graph
  - nodes = states of the component
  - s= initial state →a sourceless edge incident on s
  - a reaction →an edge from node s to node t labeled with input i and output o.
- Such graphs are called Mealy machines.
- Executions of the component =paths through this graph starting at an initial state.
- The Mealy machine representation of the Delay component

$s \xrightarrow{i/o} t.$





$0 \xrightarrow{0/0} 0;\quad 0 \xrightarrow{1/0} 1;\quad 1 \xrightarrow{0/1} 0;\quad 1 \xrightarrow{1/1} 1.$

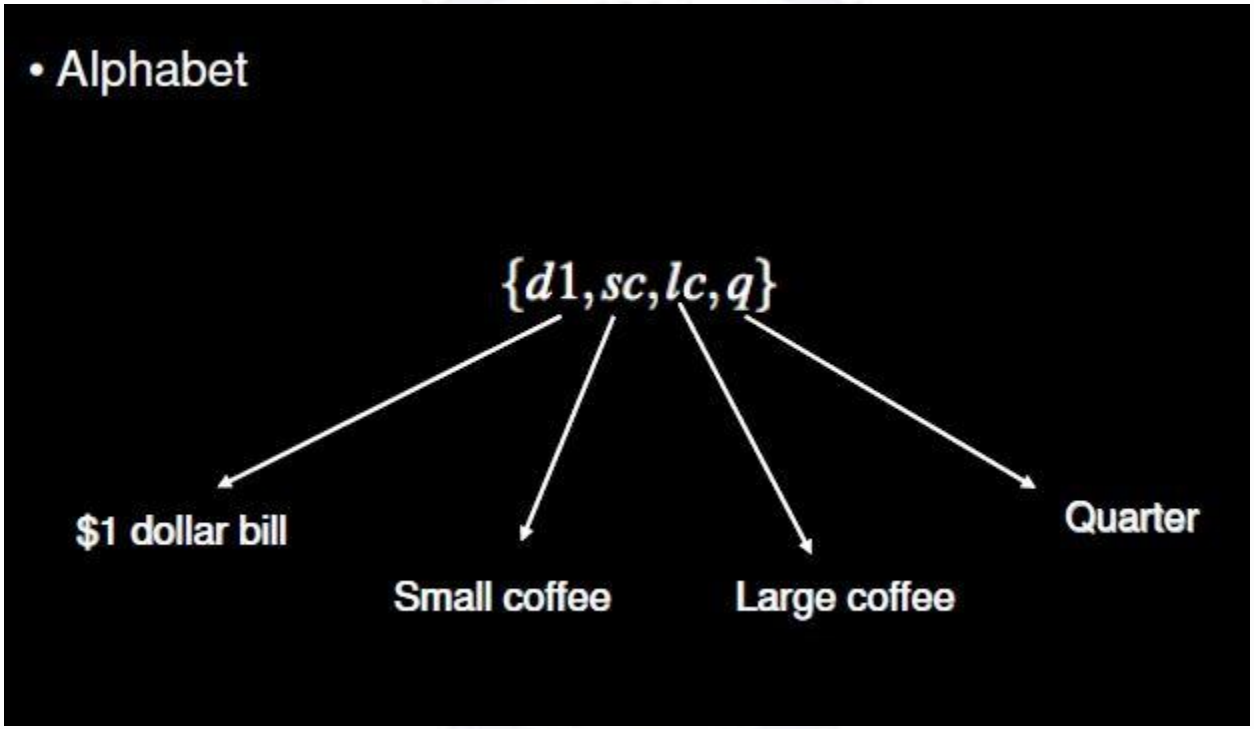
The vending machine
- Example -

- A machine that sells coffee
  - Accepts one dollar (d1) bills
  - Maximum two dollars
  - Quarters change
  - Sells two products
  - Small coffee for $1
  - Large coffee for $1.25

Denotational description basics

- *Denotational descriptions are "implicit" in the sense that they describe the properties that the system must have. They often are given as a system of equalities and inequalities that must be satisfied by the system.*
- The controller is denoted by a set of traces of symbols from an alphabet
- Non all-capital letters names belong to the alphabet of a process
- Capital letters names denote processes (CTRL is the controller process)
- A process is a letter followed by a process: P = x → Q
- SKIP is a process that successfully completes execution (it does nothing, it just completes the execution)
- If P and Q are processes then Z = P ; Q is a process that behaves like P until it completes and then like Q
- If P and Q are processes then P | Q denotes a choice between P and Q

Vending machine description



Vending machine description

- Vending machine process

$$VM = (SMALL|LARGE);VM$$

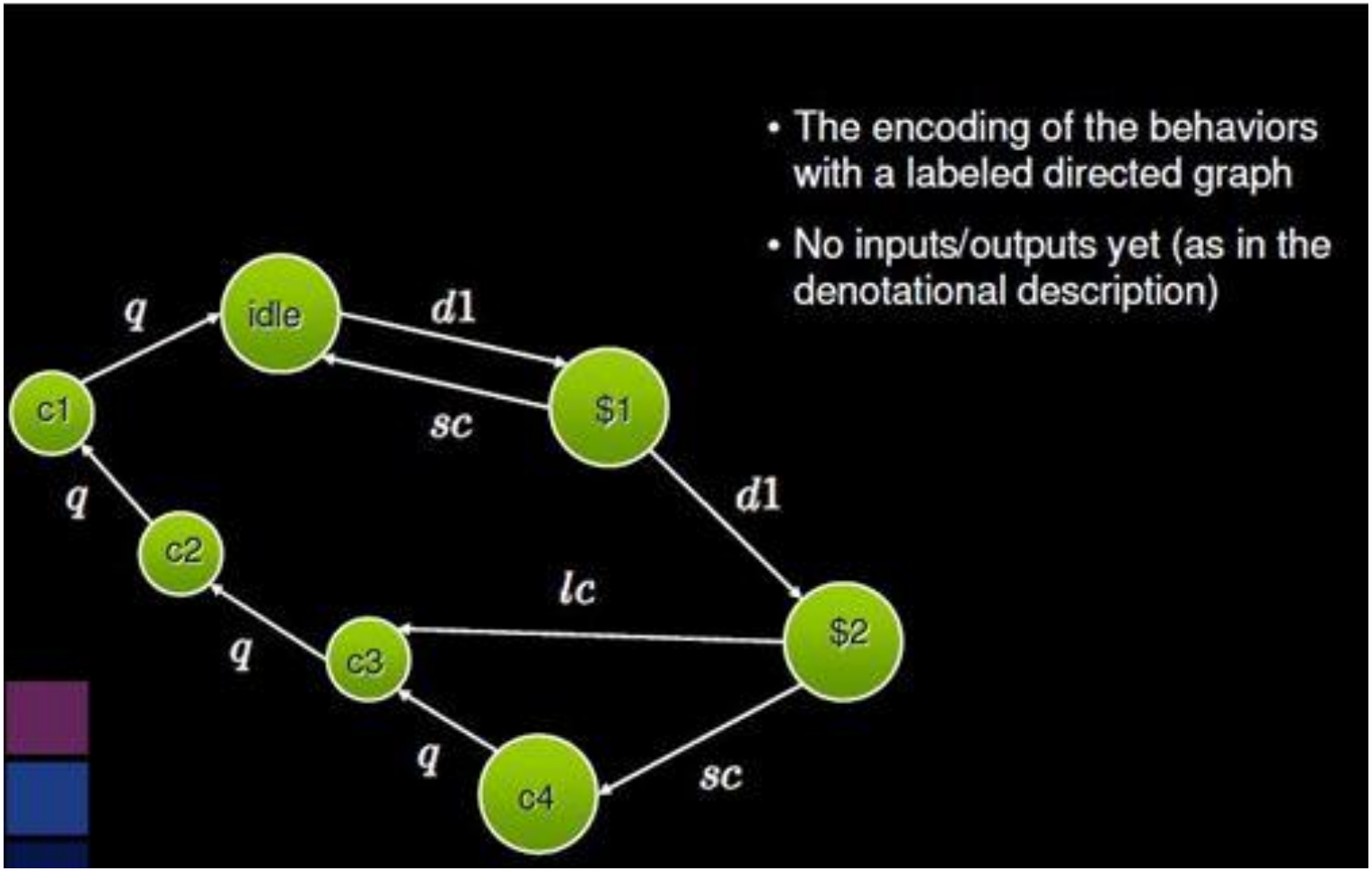
Behaves as (small "choice" large) until successful completion and then like VM

- It is a recursive definition of the form $X = F(X)$
- For a large coffee:

$$LARGE = d1 \rightarrow (d1 \rightarrow (lc \rightarrow CHANGE3))$$

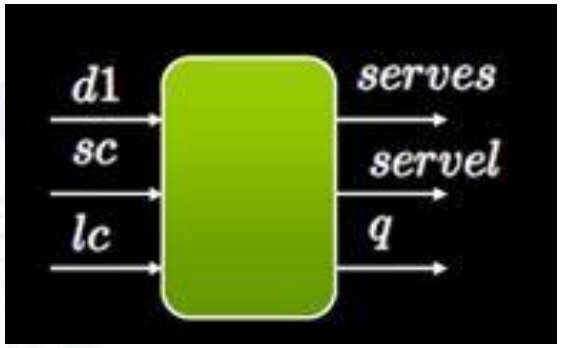
$$CHANGE3 = q \rightarrow (q \rightarrow (q \rightarrow STOP))$$

Vending machine FSM



Vending machine
I/O description

- I = {d1,sc,lc}
- O = {serves, servel, q}
- S = {idle, $1, $2, c1,c2,c3,c4}
  - (deterministic description)
    - δ : IxS→S state transition function
    - λ :IxS→O output function



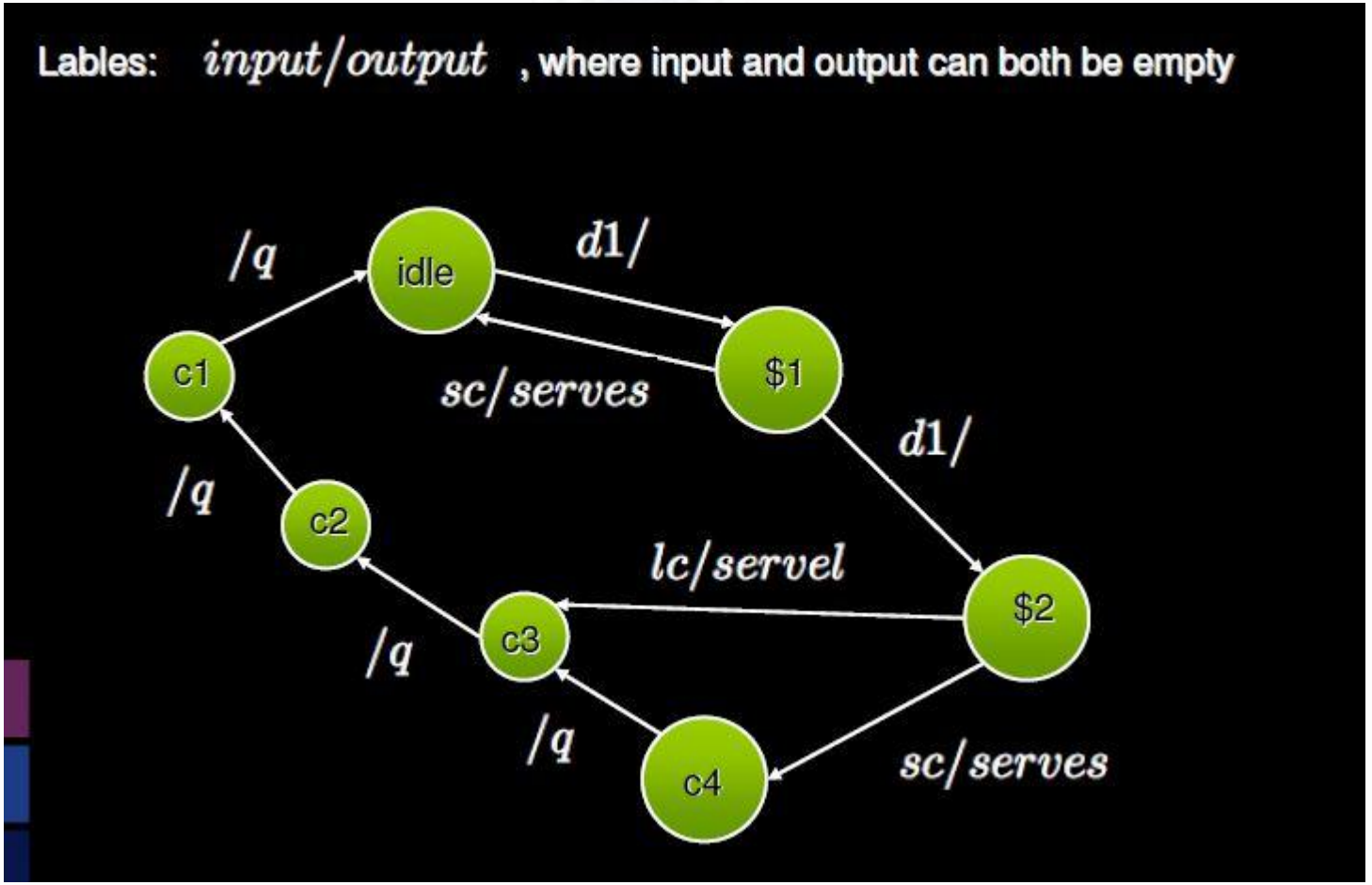
- Examples: δ(d1, idle)=$1 → if waiting and one dollar is inserted

change state to $1 credit

δ(sc, $1) = idle → if $1 credit and small coffee is requested, change state to idle and serve the coffee

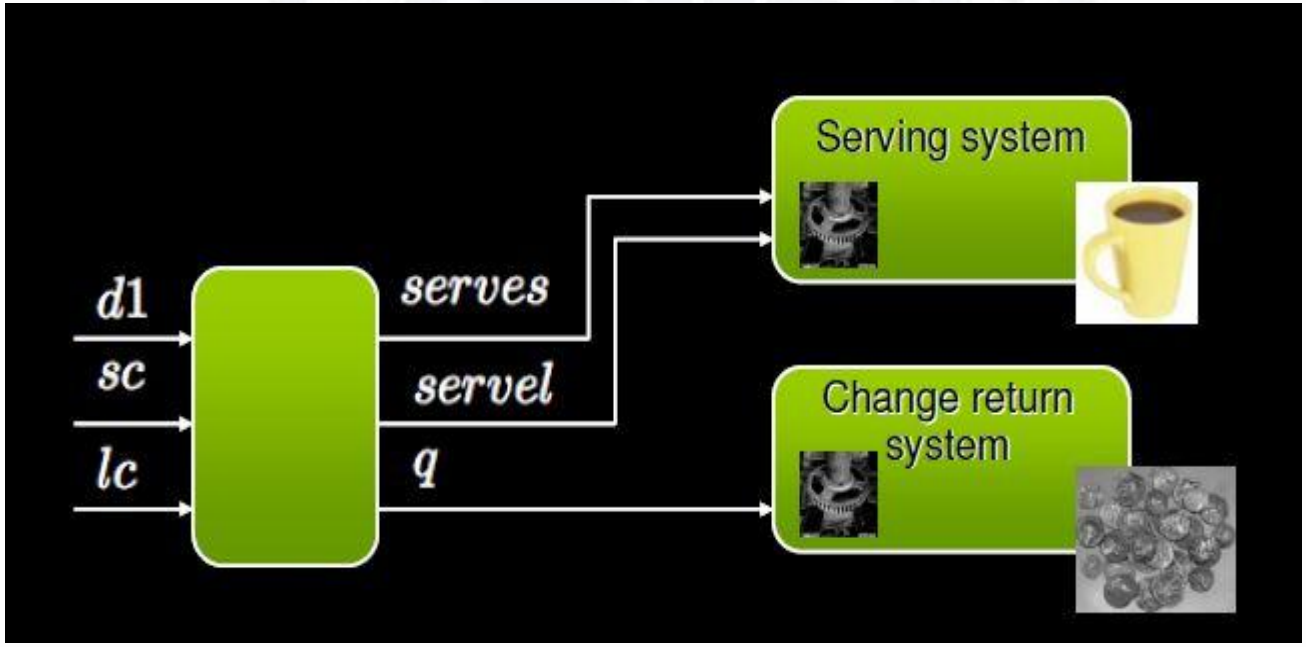
λ(sc,$1)=serves

Vending machine
I/O description



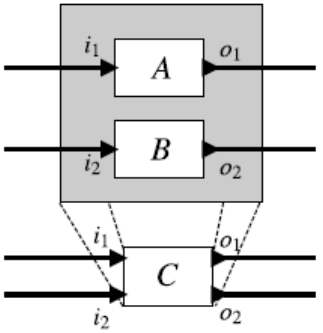
Communication with the rest of the system

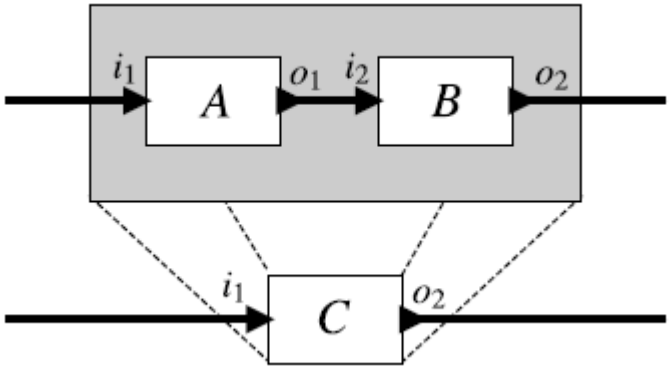
- Our state machine does not live in isolation
- – What is the communication semantics?
- – The serving system and the change return are electromechanical system with their own evolution dynamics



FSM Composition

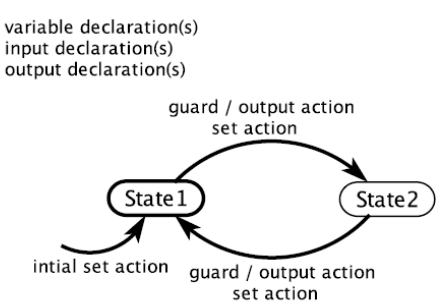
- Concurrent Composition
  - Side-by-Side Synchronous Composition
  - Side-by-Side Asynchronous Composition
  - Shared Variables
  - Cascade Composition





Extended State Machines

- The notation for FSMs becomes awkward when the number of states gets large.
- An extended state machine solves this problem by augmenting the FSM model with variables that may be read and written as part of taking a transition between states

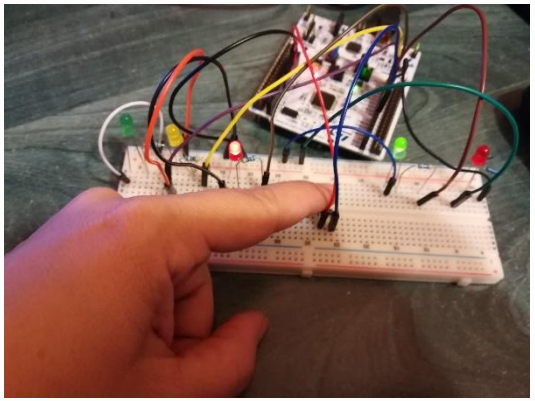


- Lee and Seshia, *Introduction to Embedded Systems*— A Cyber-Physical Systems Approach — Second Edition — MIT Press — 2017, https://ptolemy.berkeley.edu/books/leeseshia/

FSM – Demo - Traffic-Pedestrian

synchronous cascade composition

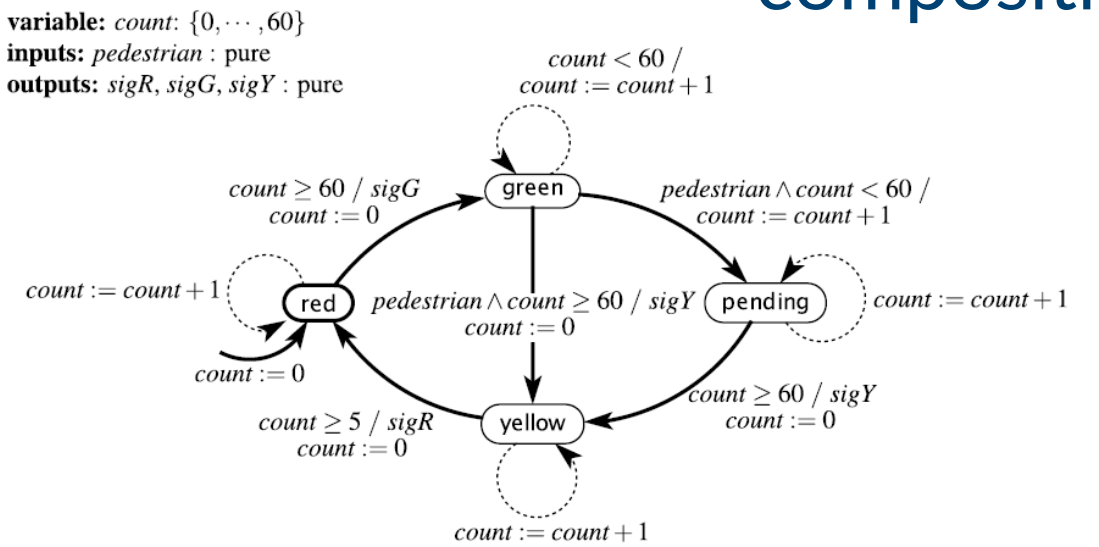
- FSM – Traffic
- FSM – Pedestrian
- FSM - Traffic – Pedestrian – Composition

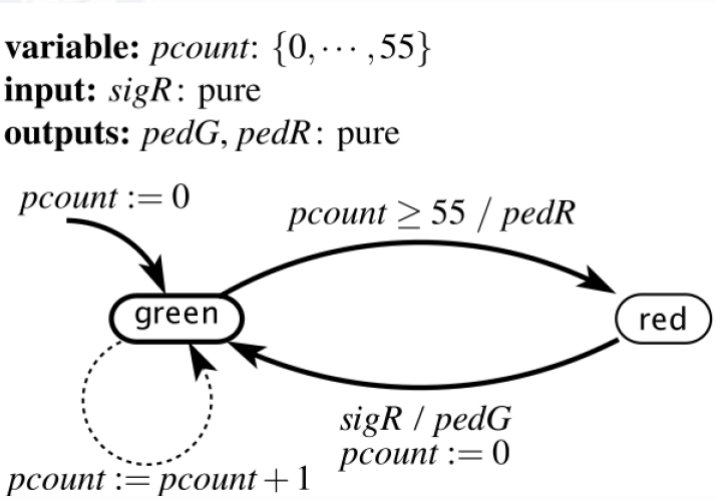


- Traffic-Pedestrian – Electronic Circuit
- Traffic-Pedestrian – Nucleo – project (video)
- Traffic-Pedestrian – Model checking - JSpin

FSM - Traffic - Pedestrian

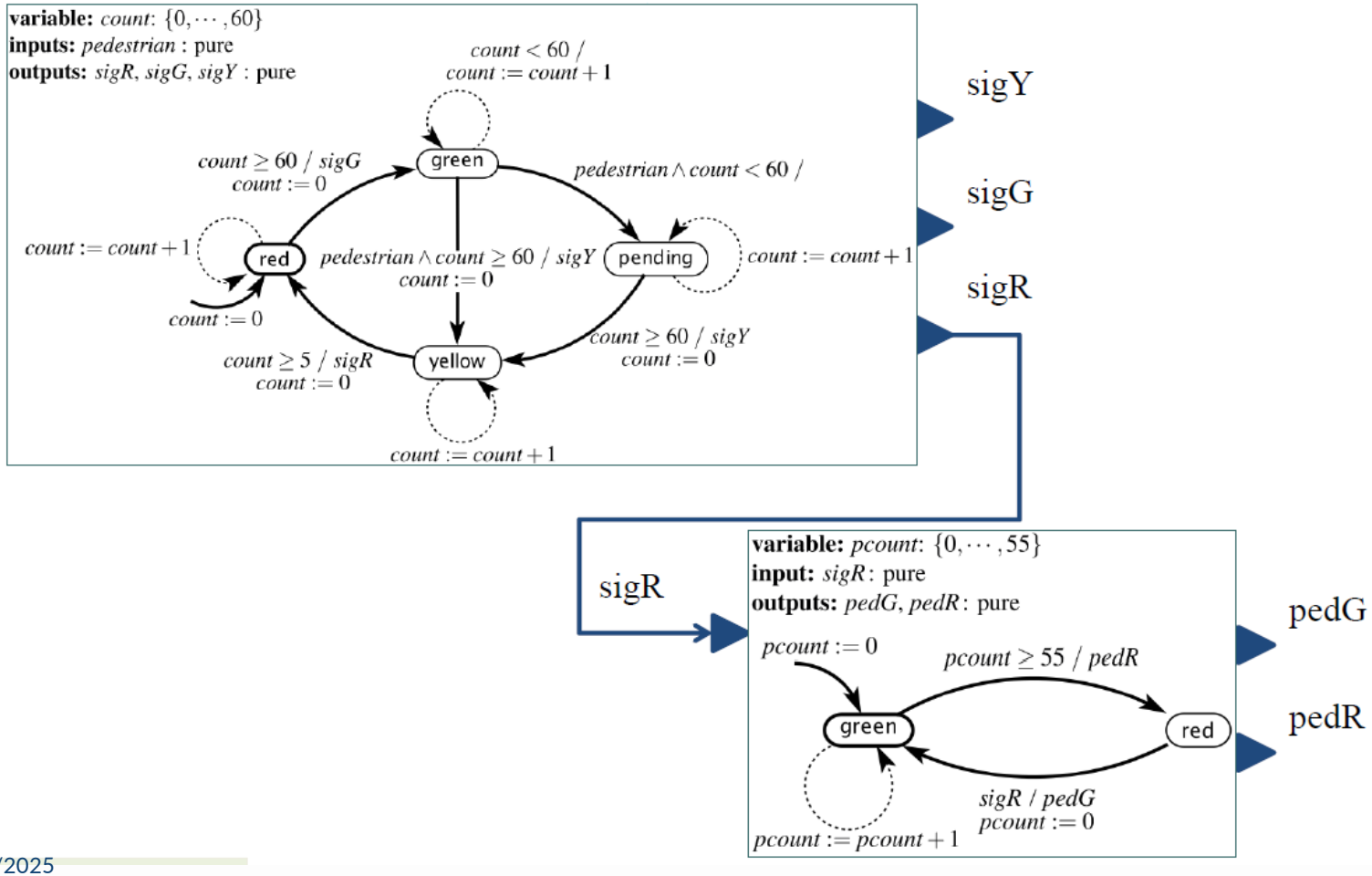
synchronous cascade composition





FSM – Traffic – Pedestrian – Composition

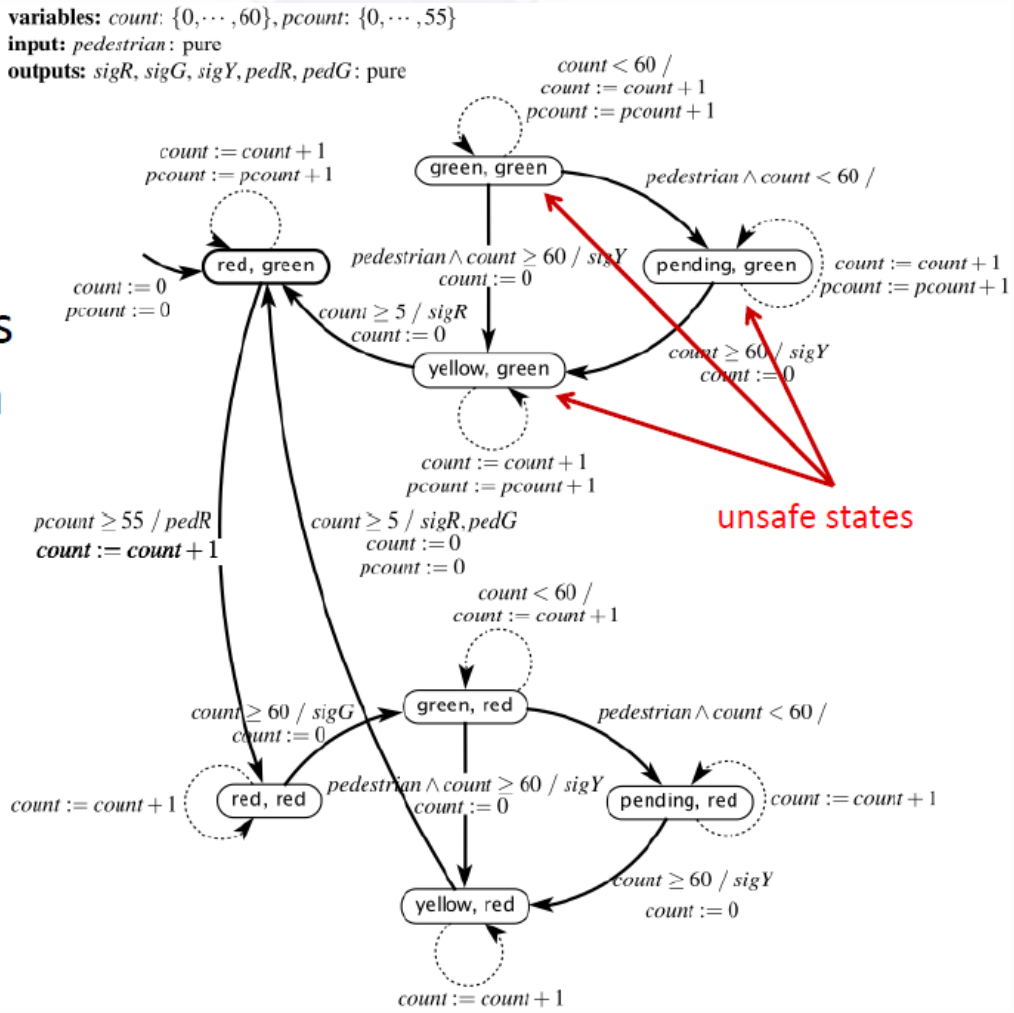
synchronous cascade composition



FSM – Traffic – Pedestrian – Composition

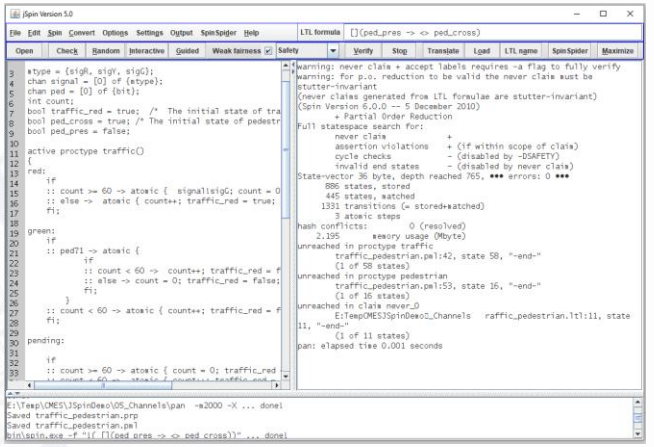
synchronous cascade composition

Synchronous composition



FSM – Demo - Traffic-Pedestrian

synchronous cascade composition



- FSM – Traffic
- FSM – Pedestrian
- FSM - Traffic – Pedestrian – Composition

- Traffic-Pedestrian – Electronic Circuit
- Traffic-Pedestrian – Nucleo – project (video)
- Traffic-Pedestrian – Model checking - JSpin

References

- Rajeev AlurPrinciples of Embedded Computation, Rajeev Alur
  - http://www.seas.upenn.edu/~cis540/
  - Rajeev Alur, Principles of Cyber-Physical Systems, 2015 - COTA:9491
  - https://www.bcucluj.ro/ro/despre-noi/filiala/biblioteca-de-matematic%C4%83-%C5%9Fi-informatic%C4%83)
- **Lee and Seshia, *Introduction to Embedded Systems*— A Cyber-Physical Systems Approach — Second Edition — MIT Press — 2017, https://ptolemy.berkeley.edu/books/leeseshia/**
- **Design of Embedded Systems: Models, Validation and Synthesis,** Alberto Sangiovanni-Vincentelli

  https://inst.eecs.berkeley.edu/~ee249/fa07/
- **System Modelling (IL2202),** Axel Jantsch

  https://www.kth.se/student/kurser/kurs/IL2202?l=en

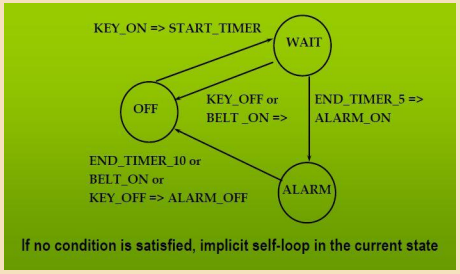
- Statecharts: a visual formalism for complex systems, David Harel, **Science of Computer Programming, 1987**
- **The STATEMATE semantics of Statecharts, David HAREL, Amnon NAAMAD, ACM, 1996**

CMES – Today

Bring it All Together

Finite State Machines

FSM – definition
+ example



Advantages

Disadvantages

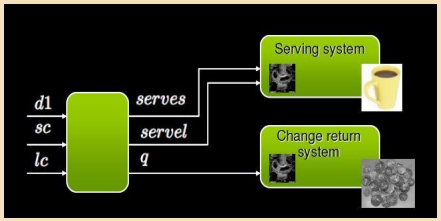
FSM composition

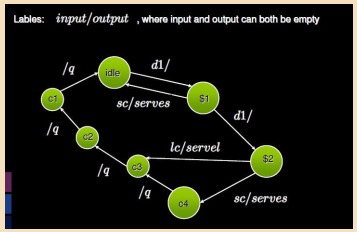
- **Bridle complexity via hierarchy:**
  FSM product yields an FSM
- **Fundamental hypothesis:**
  All the FSMs change state together (synchronicity)
- **System state** = Cartesian product of component states
  State explosion may be a problem

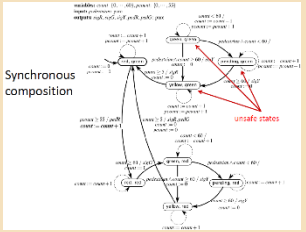
Moore vs. Mealy

The vending machine

FSM – Traffic – Pedestrian – Composition







Next Lecture

- Invited lecture
- Accenture

Thank You For Your Attention!

- ExitTicket
- Mentimeter
  - menti.com
  - Code: ?

Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

(Benjamin Franklin)