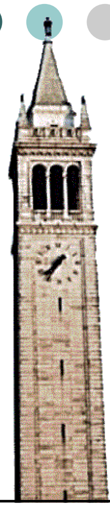


Introduction to Embedded Systems

Edward A. Lee & Sanjit A. Seshia
UC Berkeley
EECS 124
Spring 2008

Lecture 7: Modeling Modal Behavior, Part II

Recap of the last lecture

- Finite-state machine represented as a 5-tuple
  (States, Inputs, Outputs, update, initialState)
  - determinacy, receptiveness
- Non-determinism, ND FSM
  (States, Inputs, Outputs, *possibleUpdates*, *initialStates*)
- Behavior & trace

Topic: Modeling with Finite-State Machines (FSMs)

Suppose that our only modeling formalism is the Finite-State Machine

Four questions:

- ✓ How to represent the system for:
  - ✓ Mathematical analysis
  - ✓ So that a computer program can manipulate it
- ✓ How to model its environment?
- ✓ How to represent what the system MUST do – its specification?
- ➢ How to check whether the system satisfies its specification in its operating environment?

Stuttering

What happens when there is no input to the system?

i.e. at step n, $\forall$ input signals x, $x(n)$ = absent

The FSM's state remains unchanged.

This is termed **stuttering**.

Stuttering

What happens when there is no input to the system?
i.e. at step n, ∀ input signals x, x(n) = absent

The FSM's state remains unchanged.
This is termed **stuttering**.

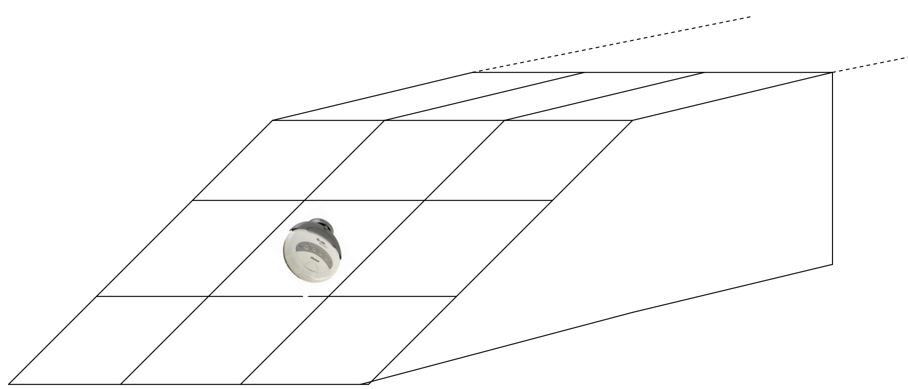
We model this using the special input symbol: **absent**

- an absent output can also be modeled this way

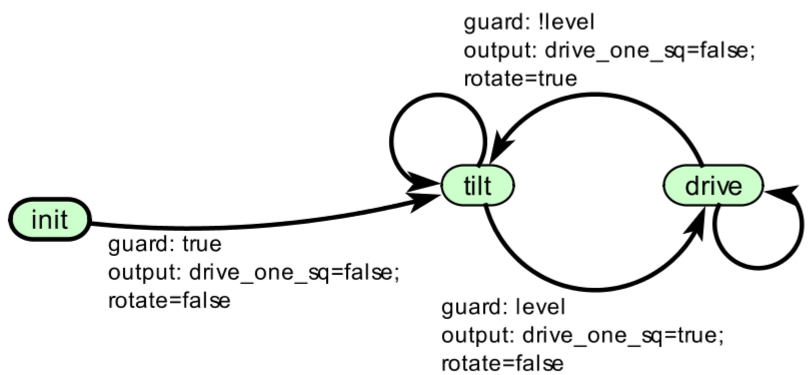
update(s, **absent**) = (s, **absent**)

- Is this the only way to model a stuttering transition?

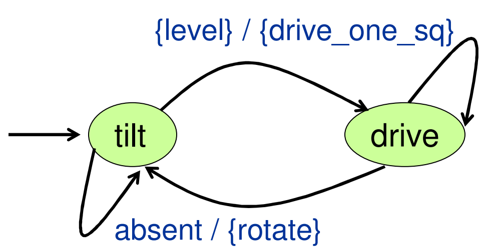
Example: Discretized iRobot Hill Climber



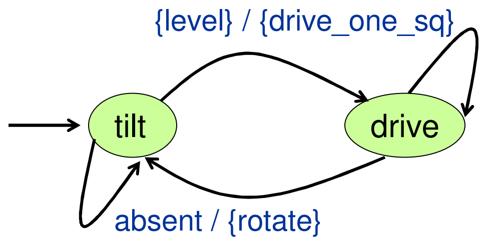
FSM Controller for iRobot



FSM Controller for iRobot

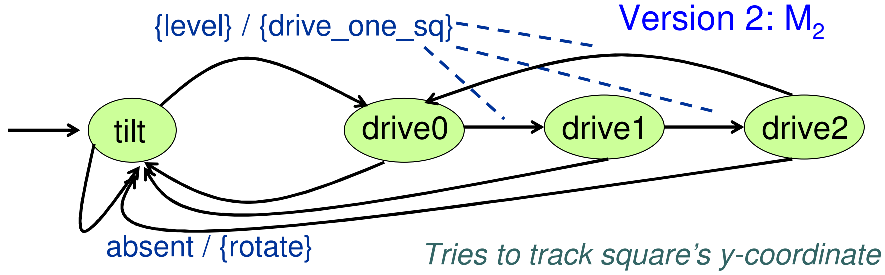


Two FSM Controllers for the iRobot



Version 1: M_1

Version 2: M_2



Tries to track square's y-coordinate

Comparing state machines

Let M_1 and M_2 be two FSMs

– M_2 obtained by implementing (adding more detail to) M_1

Mathematically, what does it mean for M_2 to implement M_1 ?

Comparing state machines

Let M_1 and M_2 be two FSMs

– M_2 obtained by implementing (adding more detail to) M_1

Mathematically, what does it mean for M_2 to implement M_1 ?

- Two possible answers:
  1. Trace containment
  2. Simulation

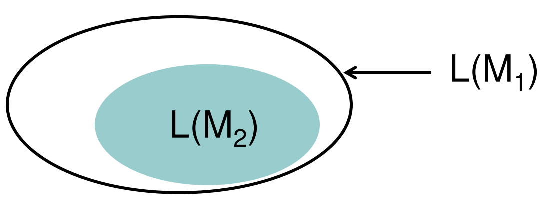
Behavior/Trace containment

Let $M_1 = (S_1, I_1, O_1, U_1, s_{10})$ and $M_2 = (S_2, I_2, O_2, U_2, s_{20})$ where $I_1 \subseteq I_2$ and $O_1 \subseteq O_2$

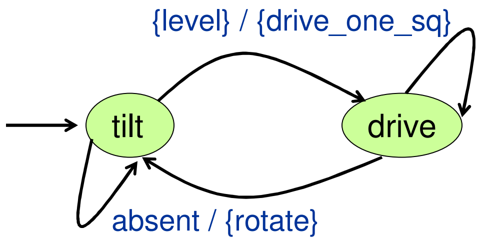
L(M) = set of *observable* traces of M

- leave out states, only retain common inputs/outputs

Defn 1: M_2 implements M_1 iff $L(M_2) \subseteq L(M_1)$

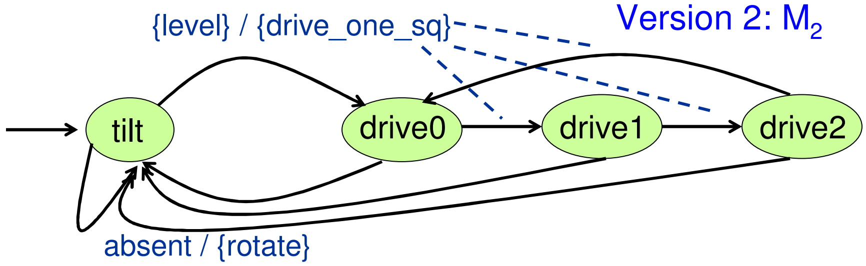


Is $L(M_2) \subseteq L(M_1)$?



Version 1: M_1

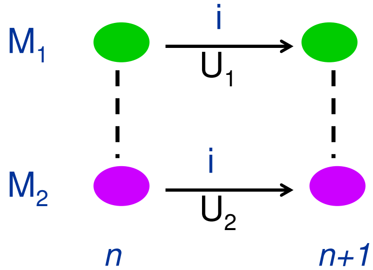
Version 2: M_2



Simulation

Intuition: a matching game between the two FSMs

Let $M_1 = (S_1, I_1, O_1, U_1, s_{10})$ and $M_2 = (S_2, I_2, O_2, U_2, s_{20})$
where $I_1 \subseteq I_2$ and $O_1 \subseteq O_2$

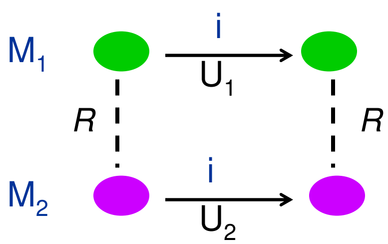


Correspondence: Common output signals take same values

Simulation

Intuition: a matching game between the two FSMs

Let $M_1 = (S_1, I_1, O_1, U_1, s_{10})$ and $M_2 = (S_2, I_2, O_2, U_2, s_{20})$
where $I_1 \subseteq I_2$ and $O_1 \subseteq O_2$



Defn2 :

M_2 implements M_1 iff
M_1 simulates M_2

Formal definition of Simulation

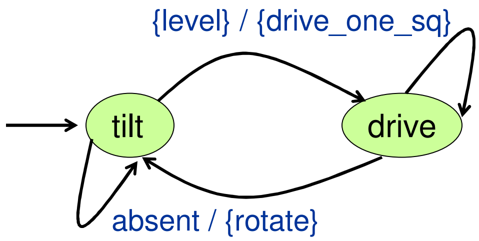
Let $M_1 = (S_1, I_1, O_1, U_1, s_{10})$ and $M_2 = (S_2, I_2, O_2, U_2, s_{20})$
where $I_1 \subseteq I_2$ and $O_1 \subseteq O_2$

We say M_1 simulates M_2 iff
there exists a set $R \subseteq S_1 \times S_2$ such that

1. $R(s_{10}, s_{20})$
2. For all $(s_1, s_2) \in R$, the following condition holds:
   For all $i \in I_2$, and $(t_2, o_2) = U_2(s_2, i)$,
   there exists a $(t_1, o_1) = U_1(s_1, i \cap I_1)$ s.t.
   $(t_1, o_1) \in R$ and $o_2 \cap O_1 = o_1$

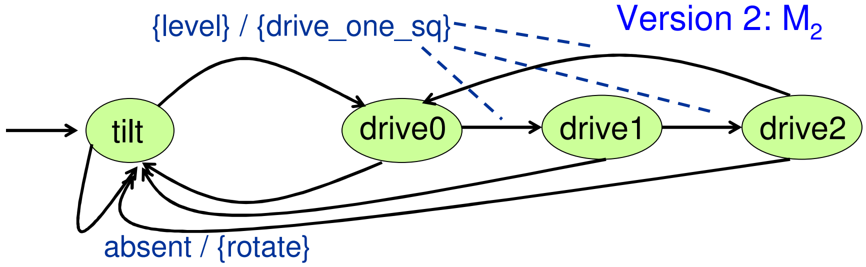
R is called the *simulation relation*

Does M_1 simulate M_2? What's R?



Version 1: M_1

Version 2: M_2



Bisimulation (differences with simulation in red)

Let $M_1 = (S_1, I_1, O_1, U_1, s_{10})$ and $M_2 = (S_2, I_2, O_2, U_2, s_{20})$
where $I = I_1 = I_2$ and $O = O_1 = O_2$

We say M_1 bisimulates M_2 iff
there exists a set $R \subseteq S_1 \times S_2$ such that

1. $R(s_{10}, s_{20})$
2. For all $(s_1, s_2) \in R$, the following conditions hold:
   For all $i \in I$, and $(t_2, o_2) = U_2(s_2, i)$,
   there exists a $(t_1, o_1) = U_1(s_1, i)$ s.t.
   $(t_1, o_1) \in R$ and $o_2 = o_1$
   For all $i \in I$, and $(t_1, o_1) = U_1(s_1, i)$,
   there exists a $(t_2, o_2) = U_2(s_2, i)$ s.t.
   $(t_2, o_2) \in R$ and $o_2 = o_1$

Simulation and Trace Containment

Theorem: If M_1 simulates M_2, then $L(M_2) \subseteq L(M_1)$

Note: If $L(M_2) \subseteq L(M_1)$ then M_1 need not simulate M_2

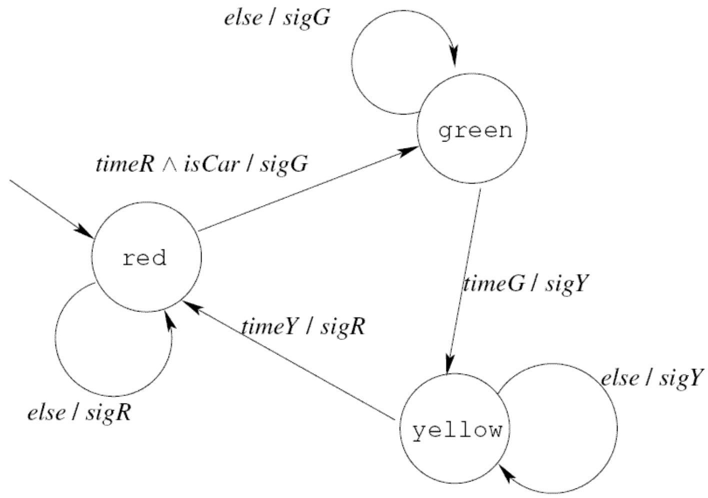
Modeling Systems with *Communicating* Finite-State Machines (FSMs)

Systems are made up of many communicating components

How do we build an FSM model from smaller component FSM models?

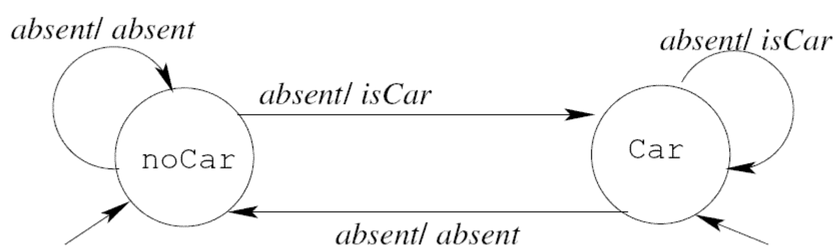
1. Composition – synchronous or asynchronous
2. Hierarchical modeling & Statecharts

Example: Traffic Light Controller



EECS 124, UC Berkeley: 21

Environment of the Controller: Cars



This FSM models the sensor input to the Controller; it has a single output signal isCar. As a modeling choice, we can introduce an input to this FSM as well.

EECS 124, UC Berkeley: 22

Composition of FSMs based on Connectivity

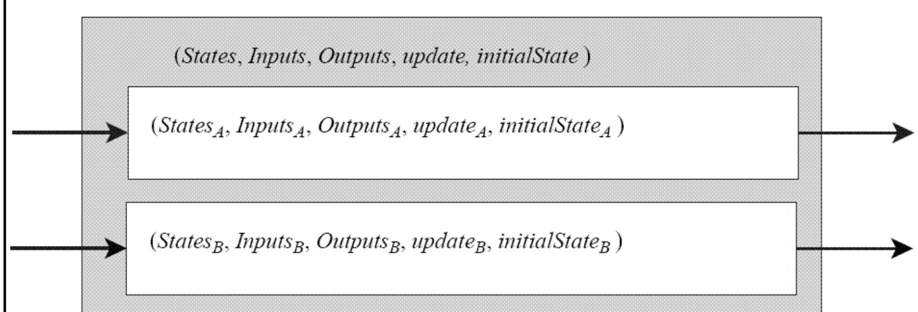
Side-by-side composition

Cascade composition

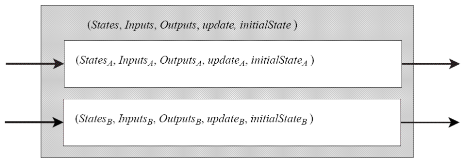
Feedback composition

(see Lee & Varaiya, Ch. 4 for more kinds of composition)

Side-by-Side Composition



Side-by-Side Composition



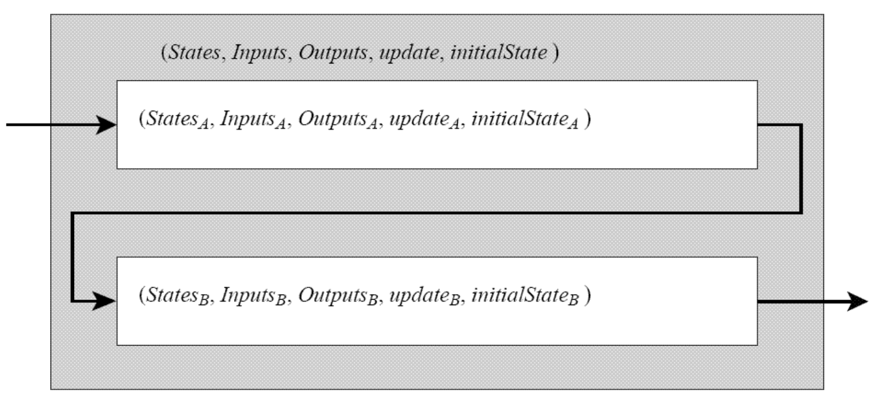
Definition of the side-by-side composite machine:

$States = States_A \times States_B$
$Inputs = Inputs_A \times Inputs_B$
$Outputs = Outputs_A \times Outputs_B$
$initialState = (initialState_A, initialState_B)$
$((s_A(n+1), s_B(n+1)), (y_A(n), y_B(n)))$
$= update((s_A(n), s_B(n)), (x_A(n), x_B(n))),$

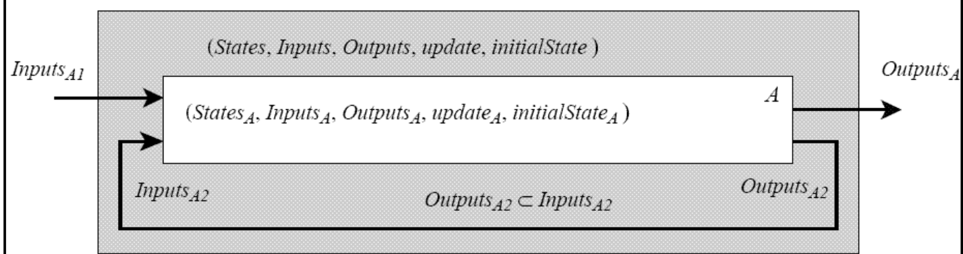
where

$(s_A(n+1), y_A(n)) = update_A(s_A(n), x_A(n))$ and
$(s_B(n+1), y_B(n)) = update_B(s_B(n), x_B(n))$

Cascade Composition

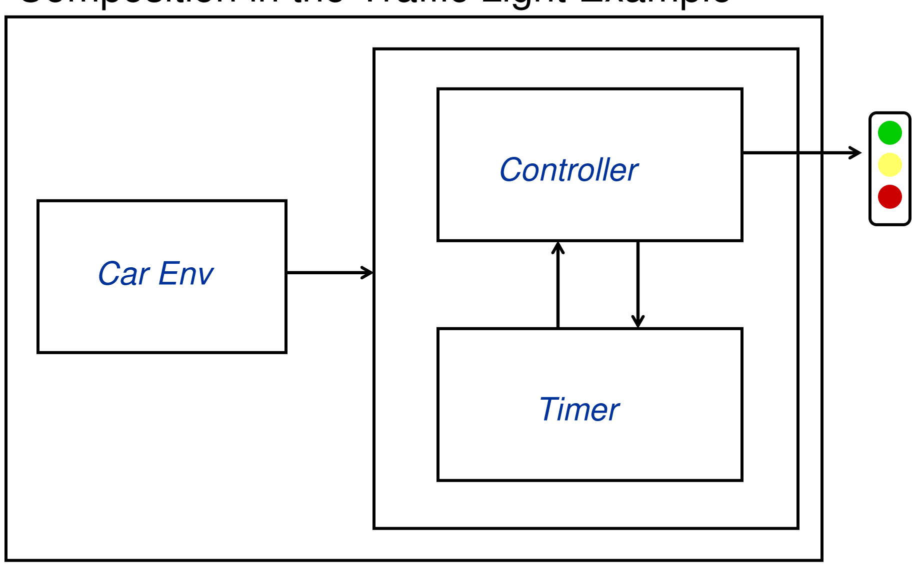


Feedback Composition



Can apply composition hierarchically

Composition in the Traffic Light Example



Synchronous Composition

$M_1 = (S_1, I_1, O_1, U_1, s_{10})$ and $M_2 = (S_2, I_2, O_2, U_2, s_{20})$

M is the synchronous composition of M_1 and M_2

=

$(S_1 \times S_2,\ I_1 \times I_2, O_1 \times O_2, U, (s_{10}, s_{20}))$

where

U is defined exactly as in the side-by-side composition definition earlier

(Note: Inputs and Outputs of M_1 and M_2 can be interconnected)

Asynchronous Composition

$M_1 = (S_1, I_1, O_1, U_1, s_{10})$ and $M_2 = (S_2, I_2, O_2, U_2, s_{20})$

M is the asynchronous composition of M_1 and M_2

$= (S_1 \times S_2,\ I_1 \times I_2, O_1 \times O_2, U, (s_{10}, s_{20}))$

where

$U((s_1, s_2), (i_1, i_2)) = ((s_1', s_2'), (o_1, o_2))$

and

$(s_1', o_1) = U_1(s_1, i_1)$ AND M_2 stutters

OR $(s_2', o_2) = U_2(s_2, i_2)$ AND M_1 stutters

The Need for Hierarchical Representation

Suppose we have a network of communicating FSMs

We want to construct and analyze the synchronous composition of all of those FSMs

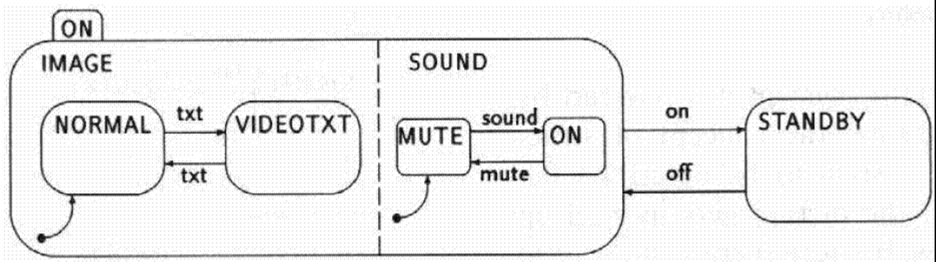
Is it easy to simply construct the composition ("flat" representation) and work with that?

Problems with Flat FSM Representation

- Exponential blow-up possible during composition
- Too much detail for humans
- Not a natural way of representing parallel composition
- Does not reflect modular (top-down or bottom-up) development
- Common structure cannot be easily extracted: e.g., what happens when an interrupt is received?

Hierarchical Modeling and Statecharts

Modeling with
- Hierarchy
- Orthogonality (AND-states and OR-states)
- Broadcast (for communication)



Example due to Reinhard von Hanxleden