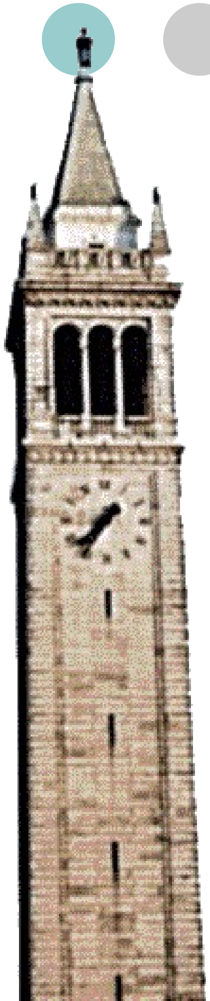


Introduction to Embedded Systems

Sanjit A. Seshia

UC Berkeley

EECS 149/249A

Fall 2015

Chapter 5: Composition of State Machines

Text by Nicolae Cleju in this color

Composition of State Machines

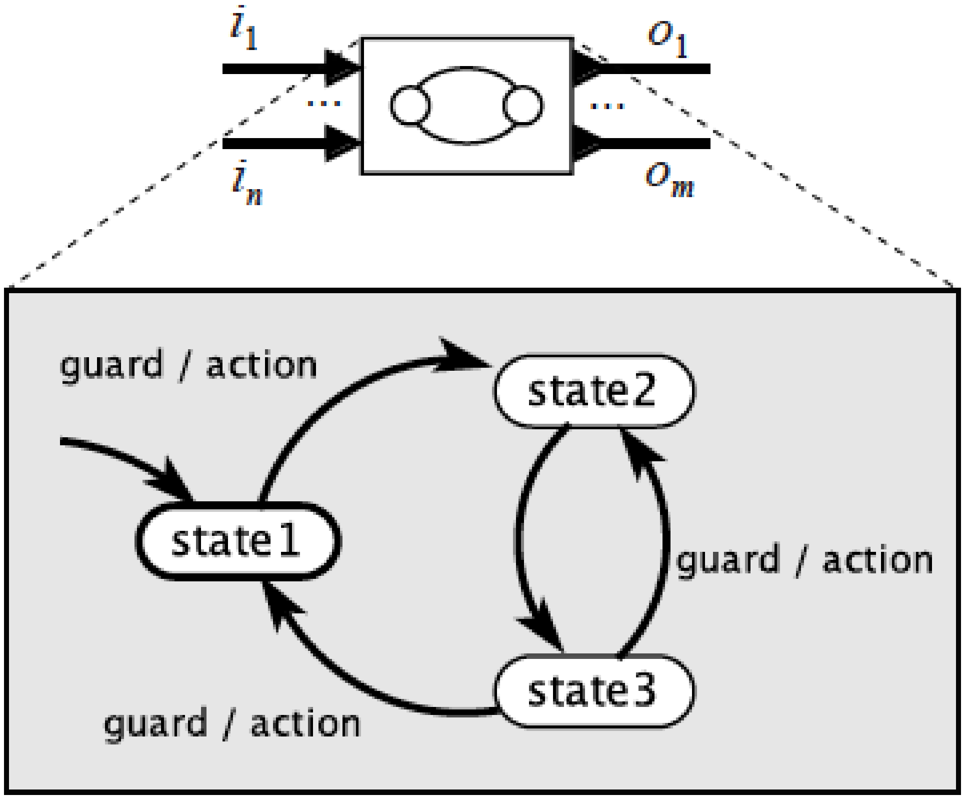
How do we construct complex state machines out of simpler “building blocks”?

Two kinds of composition:

1. **Spatial**: how do the components communicate between each other?
2. **Temporal**

Actor Model for State Machines

Expose inputs and outputs, enabling composition:



Spatial Composition of State Machines

Side-by-side composition

- No common inputs/outputs, no shared data

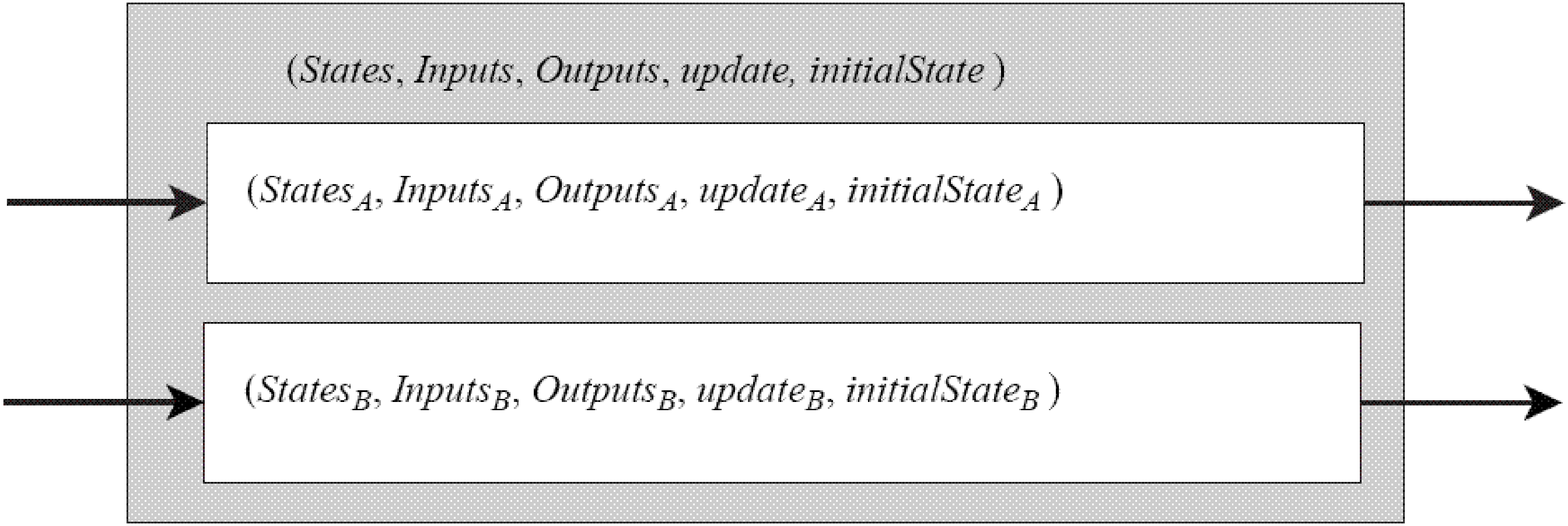
Cascade composition

- Outputs of one FSM are inputs for the second GSM

Feedback composition

- Outputs of a FSM are inputs to the same FSM (feedback)

Side-by-Side Composition



$(States, Inputs, Outputs, update, initialState)$
$(States_A, Inputs_A, Outputs_A, update_A, initialState_A)$
$(States_B, Inputs_B, Outputs_B, update_B, initialState_B)$

A key question: When do these machines react?

Temporal Composition of State Machines

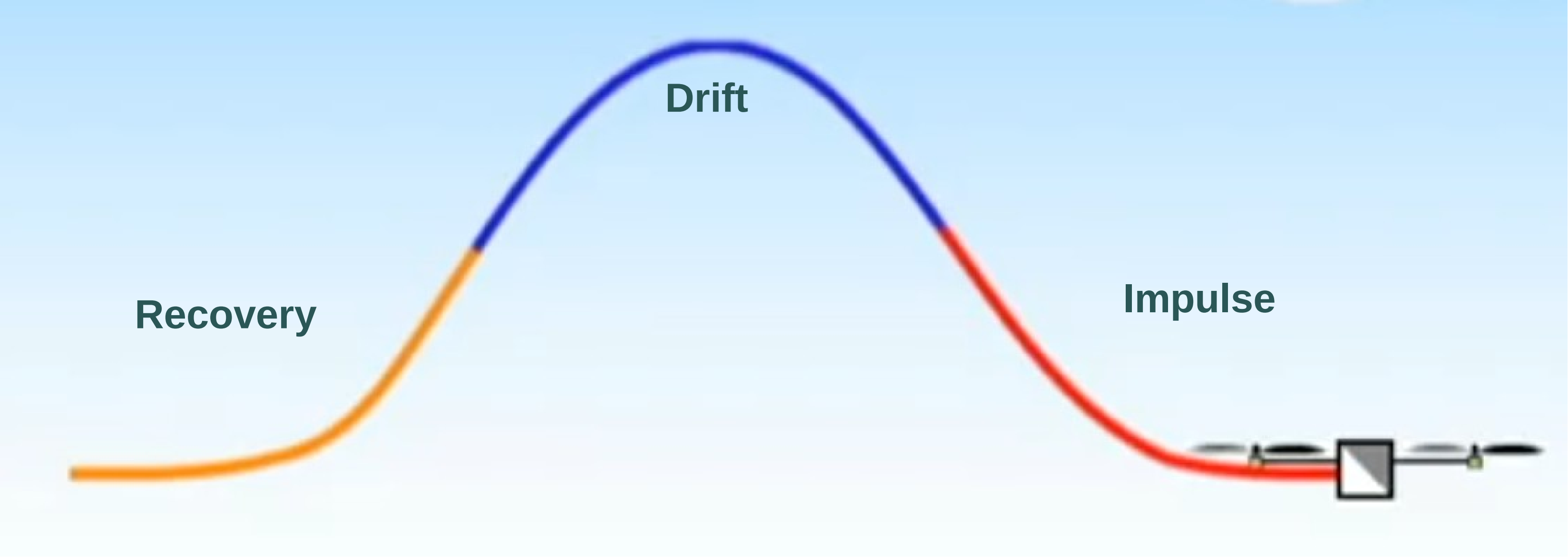
Sequential vs. Parallel

- Sequential: the two FSM do not work at the same time
- Parallel: the two FSM work at the same time

Asynchronous vs. Synchronous

- Only for parallel compositions
- Synchronous: transitions are taken at the same time in both FSMs
- Asynchronous: transitions are taken at independent times in the two FSMs

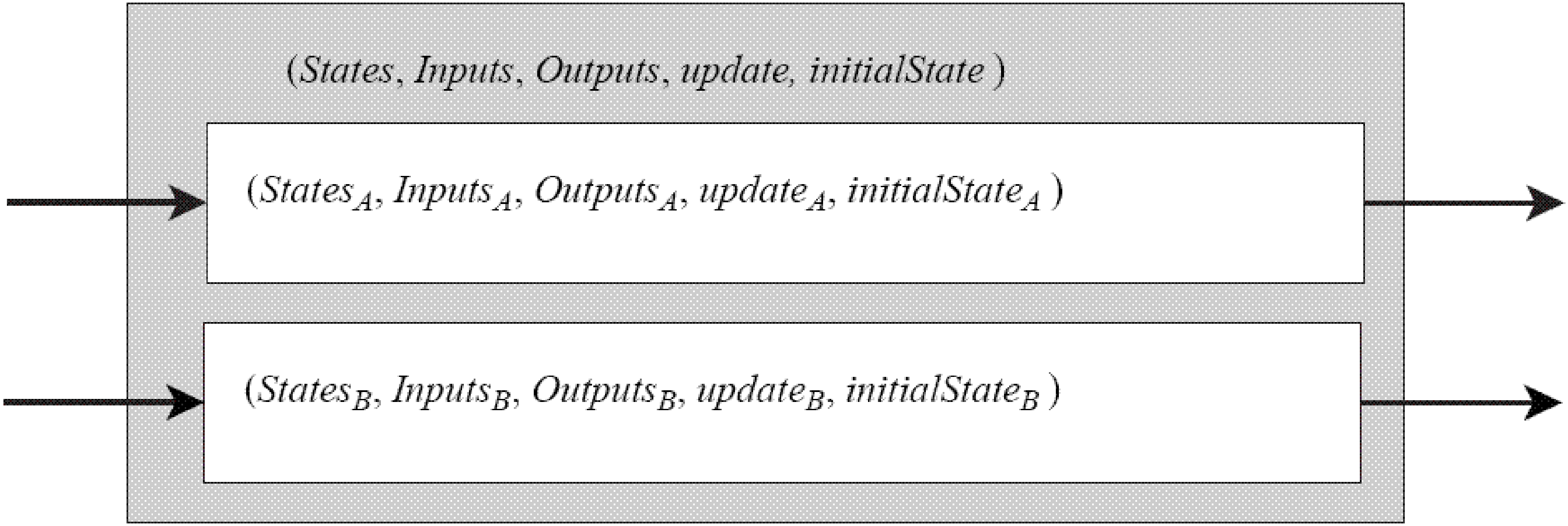
Example of Sequential Composition



https://www.youtube.com/watch?v=iD3QgGpzzIM

[Tomlin et al.]

Side-by-Side, Parallel Composition


$(States, Inputs, Outputs, update, initialState)$
$(States_A, Inputs_A, Outputs_A, update_A, initialState_A)$
$(States_B, Inputs_B, Outputs_B, update_B, initialState_B)$

When do these machines react?

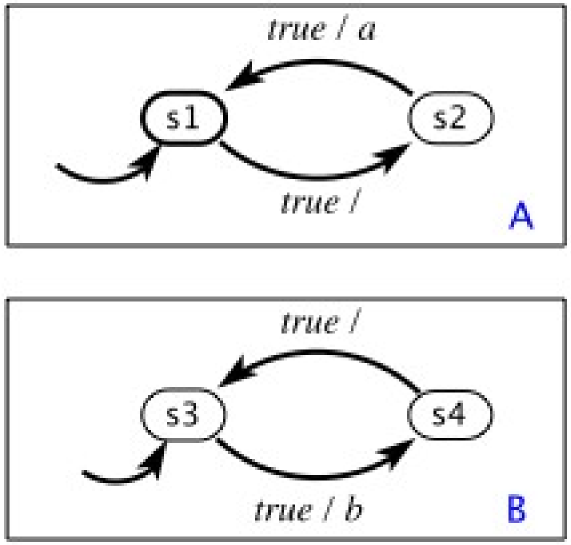
Two possibilities:

- Together, in lock step (synchronous, parallel composition)
- Independently (asynchronous, parallel composition)

Synchronous Composition

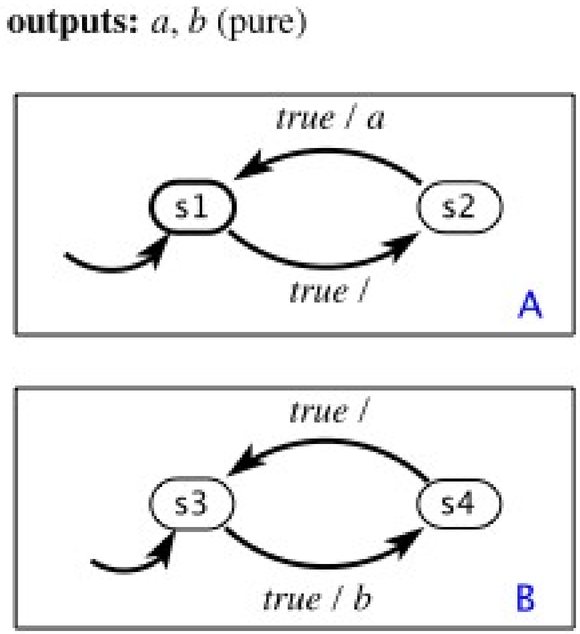
$$S_C \subseteq S_A \times S_B$$

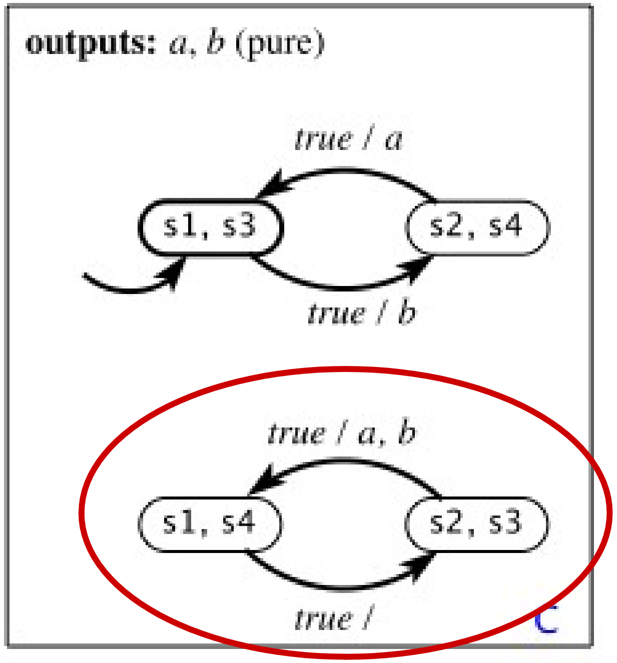
outputs: a, b (pure)



Synchronous Composition

$$S_C \subseteq S_A \times S_B$$





Synchronous composition

Note that these two states are not reachable.

Synchronous Composition

Composition details

- Composition model states = combination of states of the two FSMs

Synchronous composition

- Transition = transition in FSM A and FSM B simultaneously. Both actions happen simultaneously.
- There might exist unreachable states in the Composition model (states that will never be reached)

Asynchronous Composition

Composition details

- Composition States = combination of states of the two FSMs

Asynchronous composition

- Transitions in the two FSMs can take place at irregular and independent (not synchronized) times
- All states are reachable (can you show this?)

Asynchronous Composition

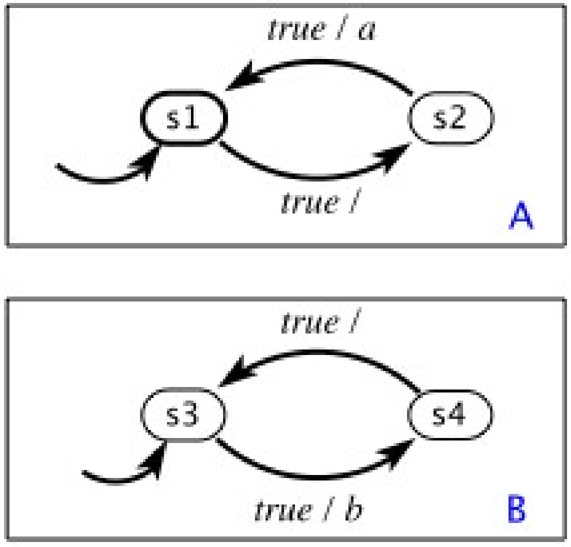
Flavors of asynchronous composition

- How are simultaneous transitions handled?
- Interleaving semantics:
  - simultaneous transition in models A and B is not allowed (we may have either a transition in model A, or a transition in B)
  - i.e. transition from A takes place first, then transition from B takes place after a non-zero time delay (or vice-versa)
- Simultaneous semantics:
  - simultaneous transition in models A and B is allowed
  - for example, we may have either
    - transition only in model A
    - transition only in model B
    - Simultaneous transition in models A and B

Asynchronous Composition

$$S_C \subseteq S_A \times S_B$$

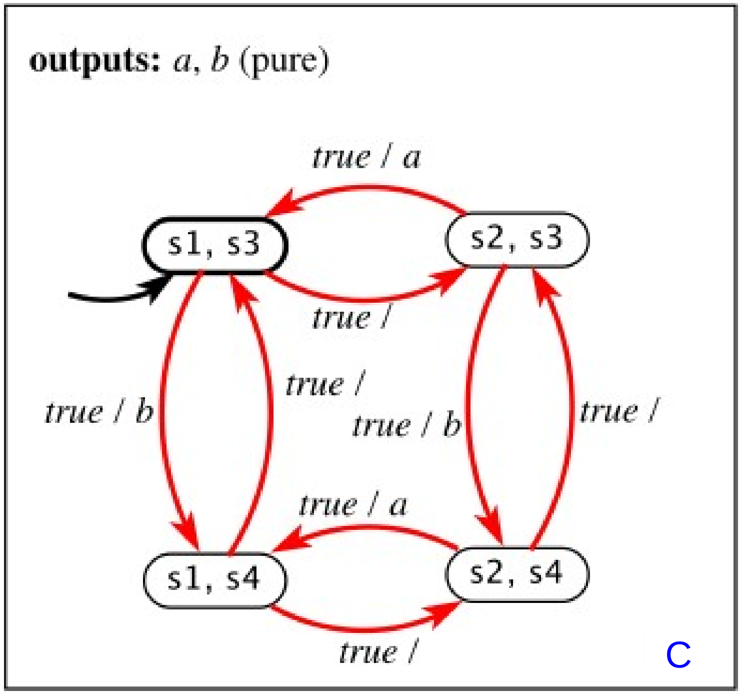
outputs: *a*, *b* (pure)



Asynchronous composition using <u>interleaving</u> semantics

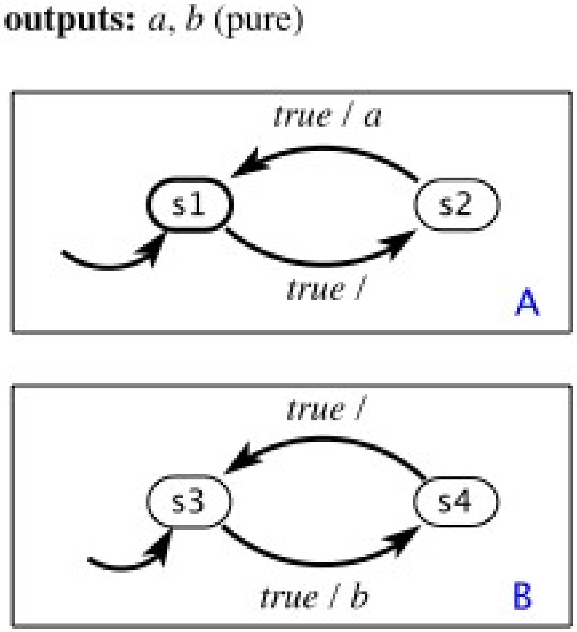
Asynchronous Composition

$$S_C \subseteq S_A \times S_B$$

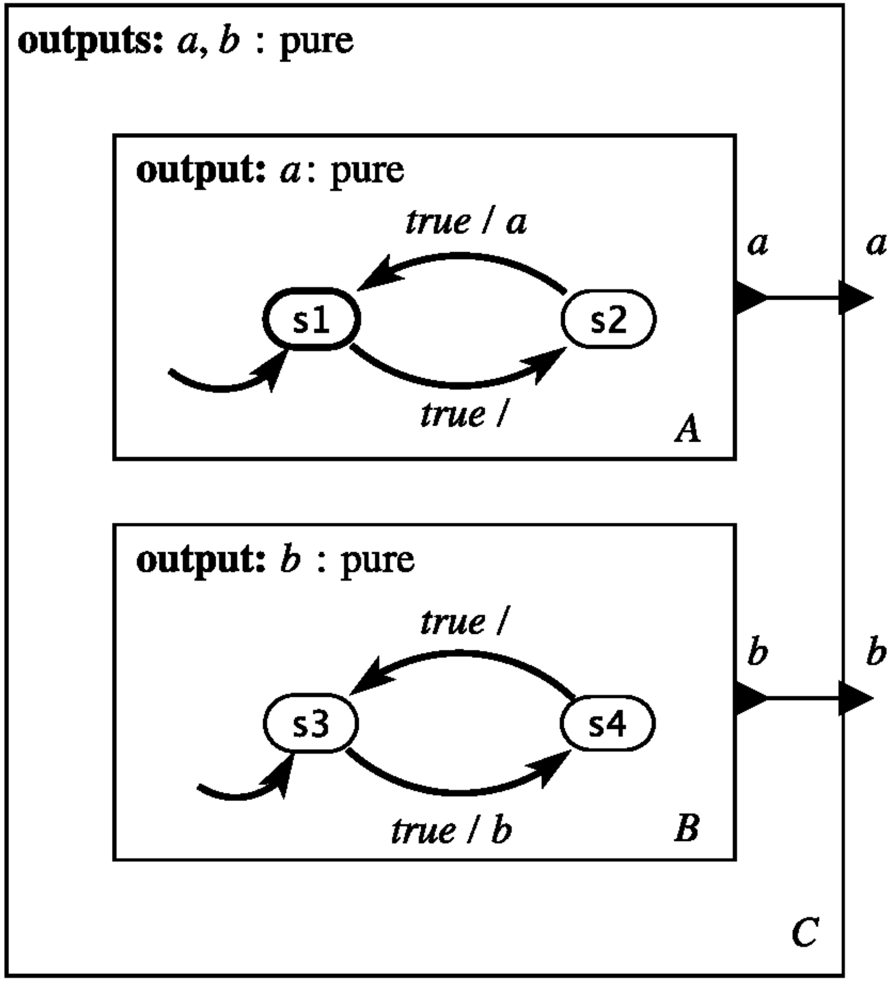


Asynchronous composition using interleaving semantics

Note that now all states are reachable.



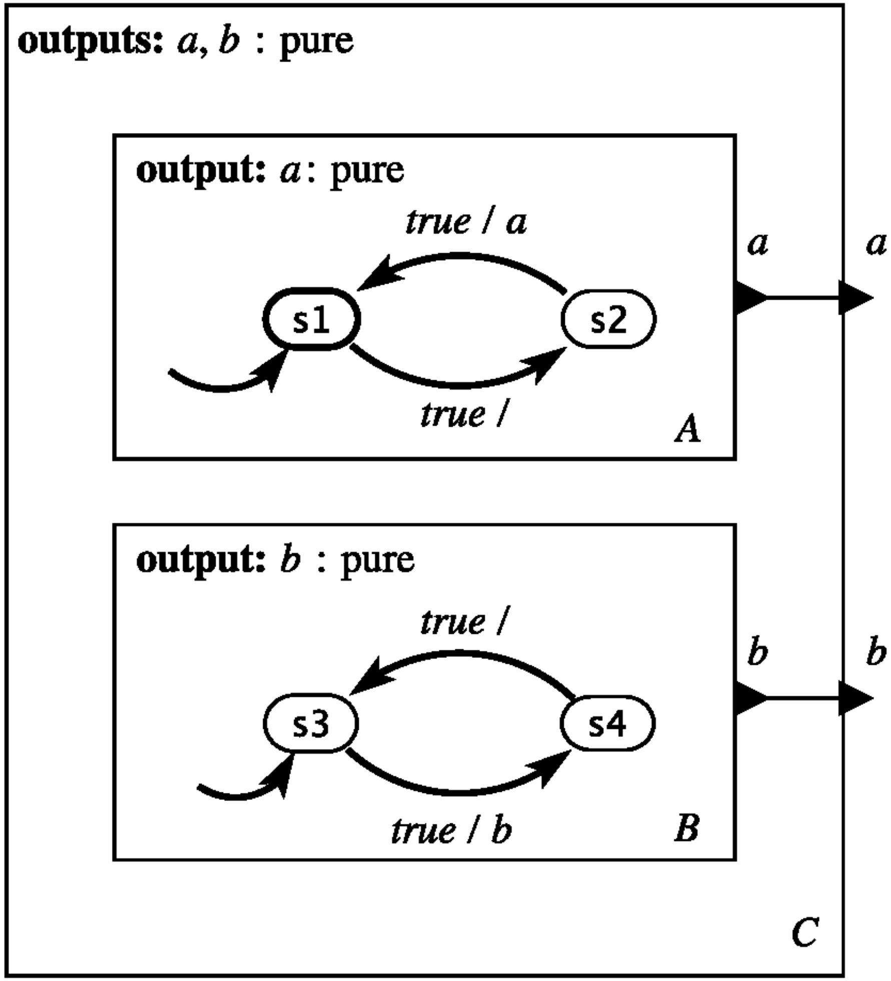
Syntax vs. Semantics



Synchronous
or
Asynchronous
composition?

If asynchronous,
does it allow
simultaneous
transitions in
A & B?

Syntax vs. Semantics

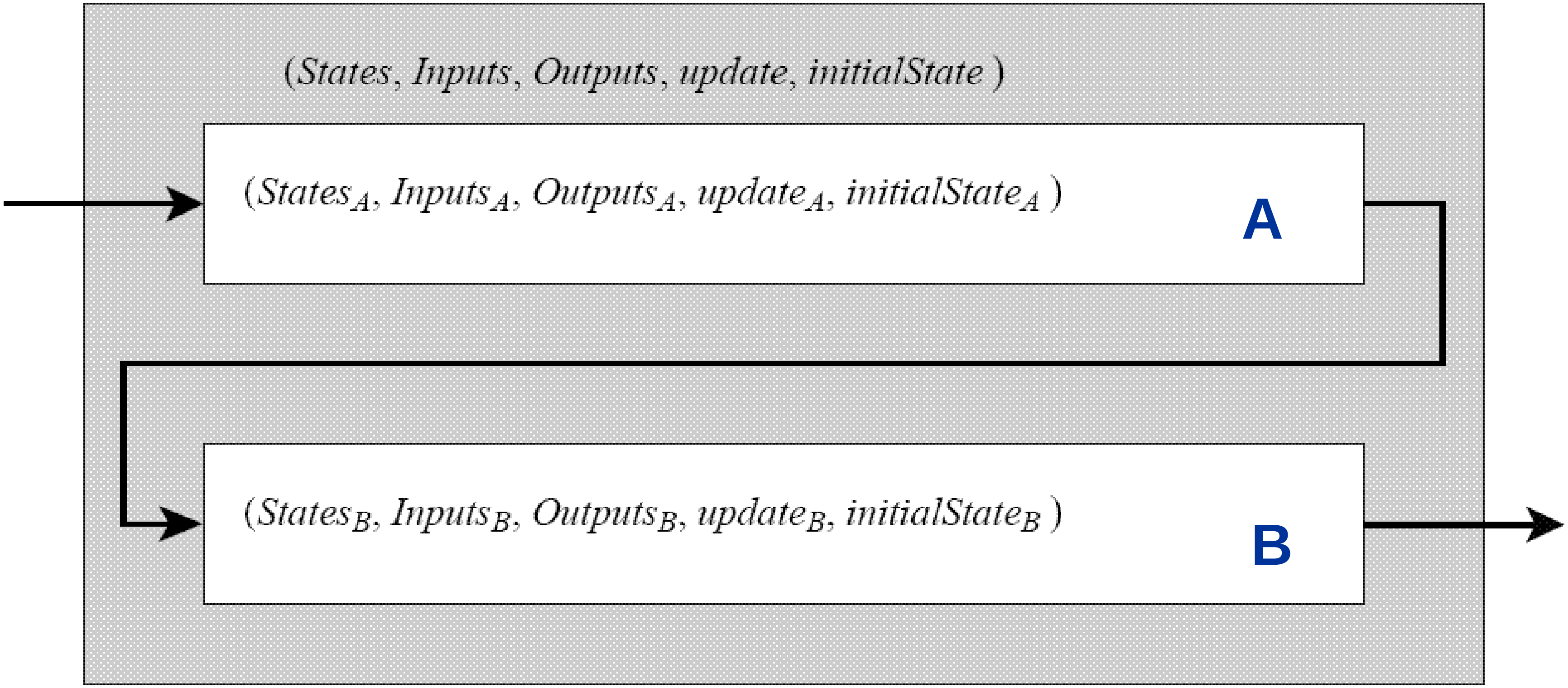


Synchronous
or
Asynchronous
composition?

If asynchronous,
does it allow
simultaneous
transitions in
A & B?

The model drawing doesn’t tell this (it represents syntax, which is different from the semantics)
This type of composition must be explained separately.

Cascade Composition

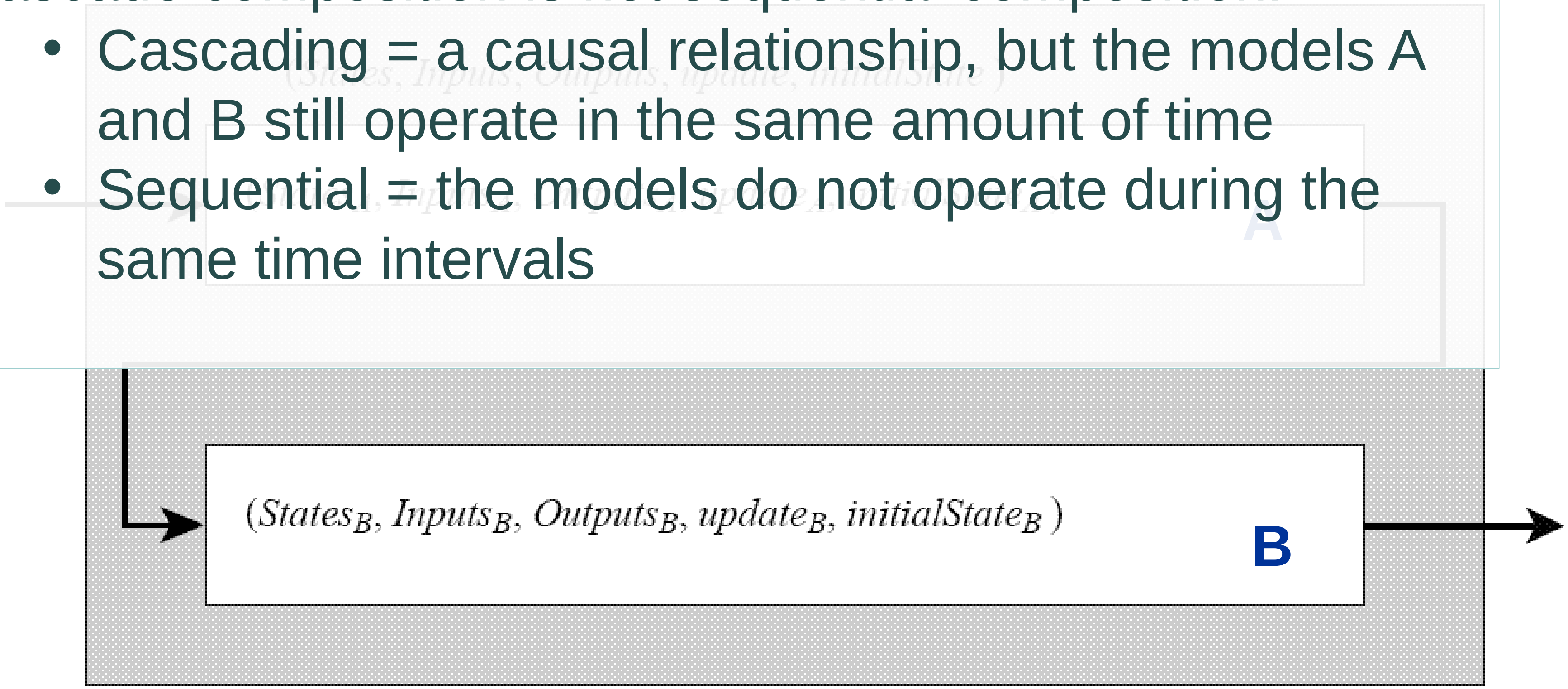

$(States, Inputs, Outputs, update, initialState)$
$(States_A, Inputs_A, Outputs_A, update_A, initialState_A)$
A
$(States_B, Inputs_B, Outputs_B, update_B, initialState_B)$
B

Output port(s) of A connected to input port(s) of B

Cascade Composition

Cascade composition is not sequential composition!

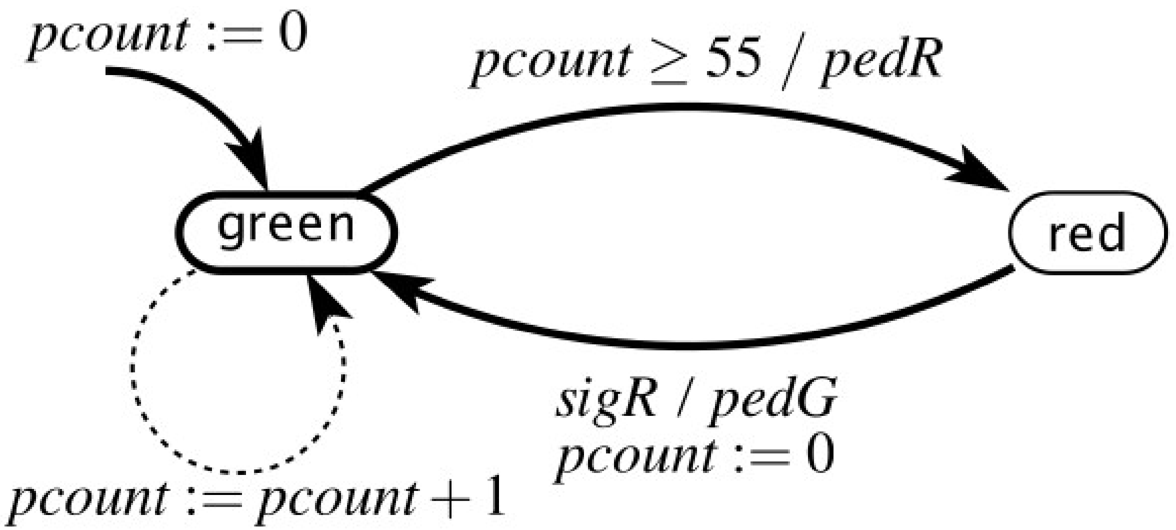
- Cascading = a causal relationship, but the models A and B still operate in the same amount of time
- Sequential = the models do not operate during the same time intervals


(States, Inputs, Outputs, update, initialState)
A
(States_B, Inputs_B, Outputs_B, update_B, initialState_B)
B

Output port(s) of A connected to input port(s) of B

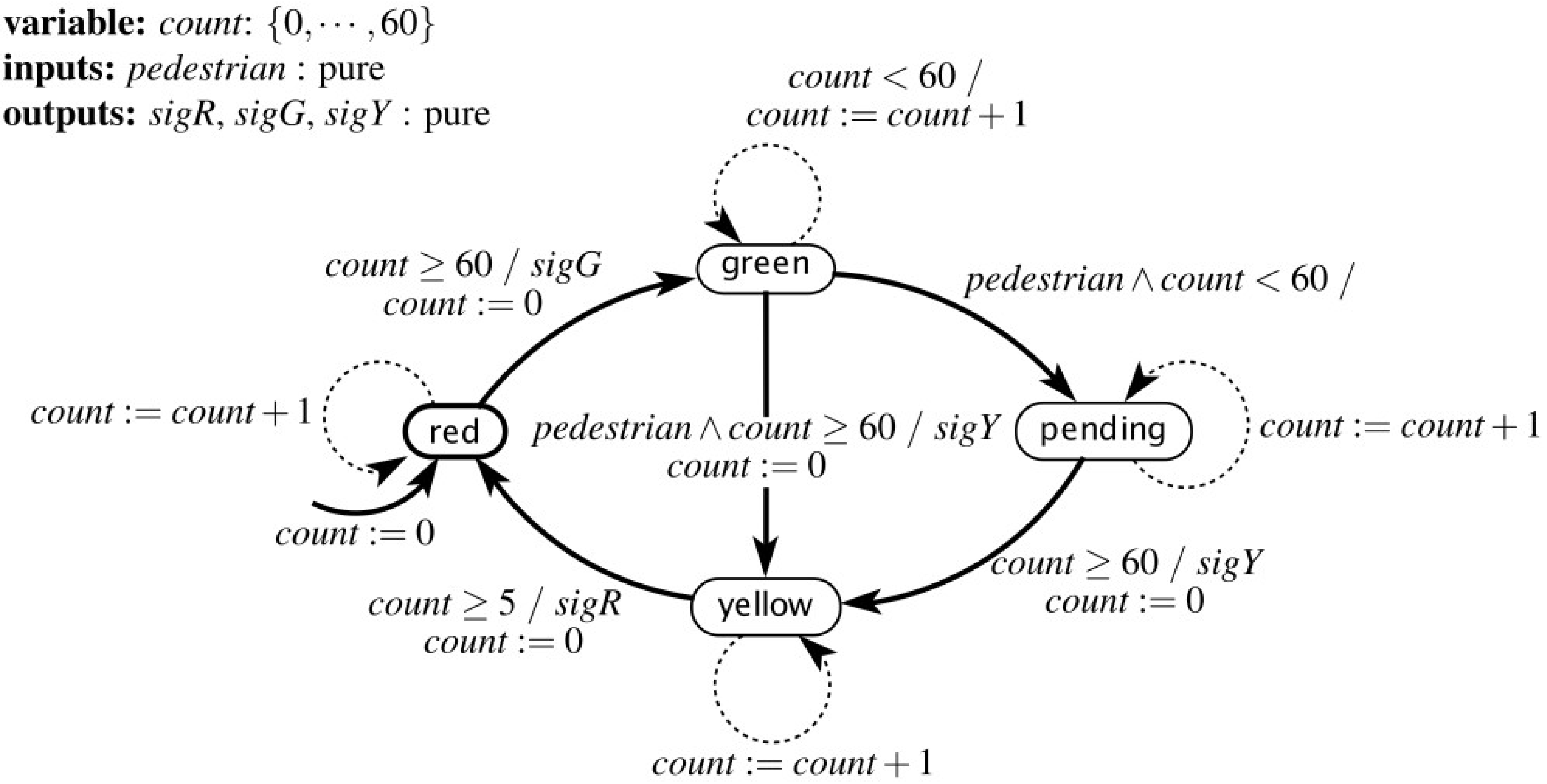
Example: Pedestrian Light

variable: $pcount\colon \{0, \cdots, 55\}$
input: $sigR\colon$ pure
outputs: $pedG, pedR\colon$ pure



This light stays green for 55 seconds, then goes red. Upon receiving a sigR input, it repeats the cycle.

Example: Car Light

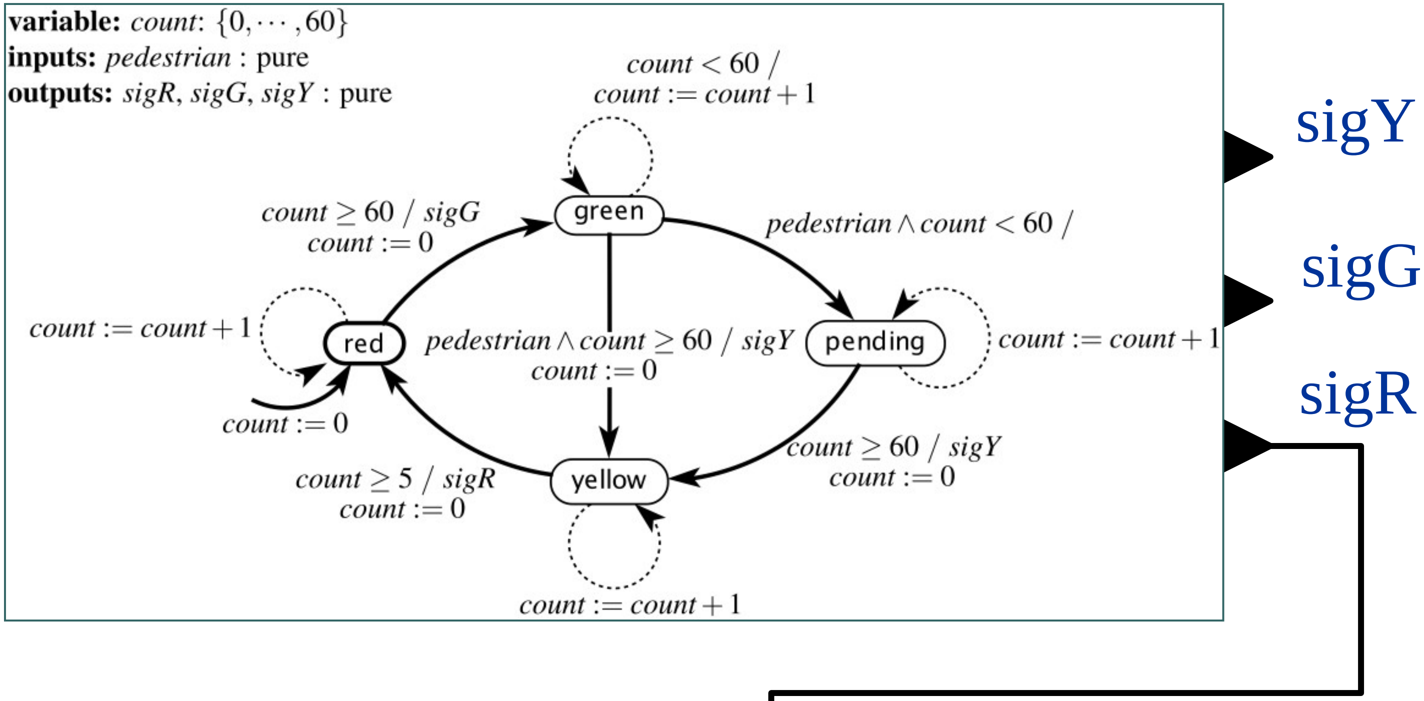


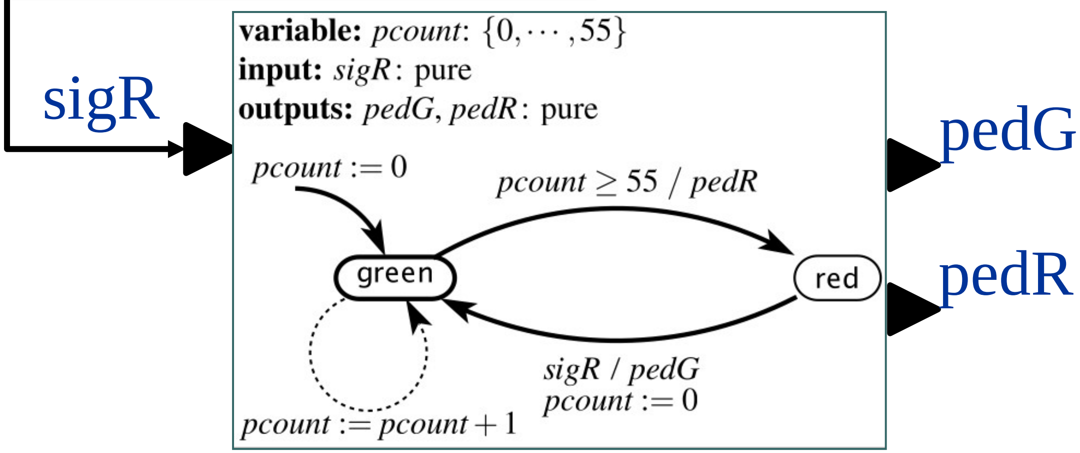
variable: $count: \{0, \cdots, 60\}$

inputs: $pedestrian$: pure

outputs: $sigR, sigG, sigY$: pure

Pedestrian Light with Car Light

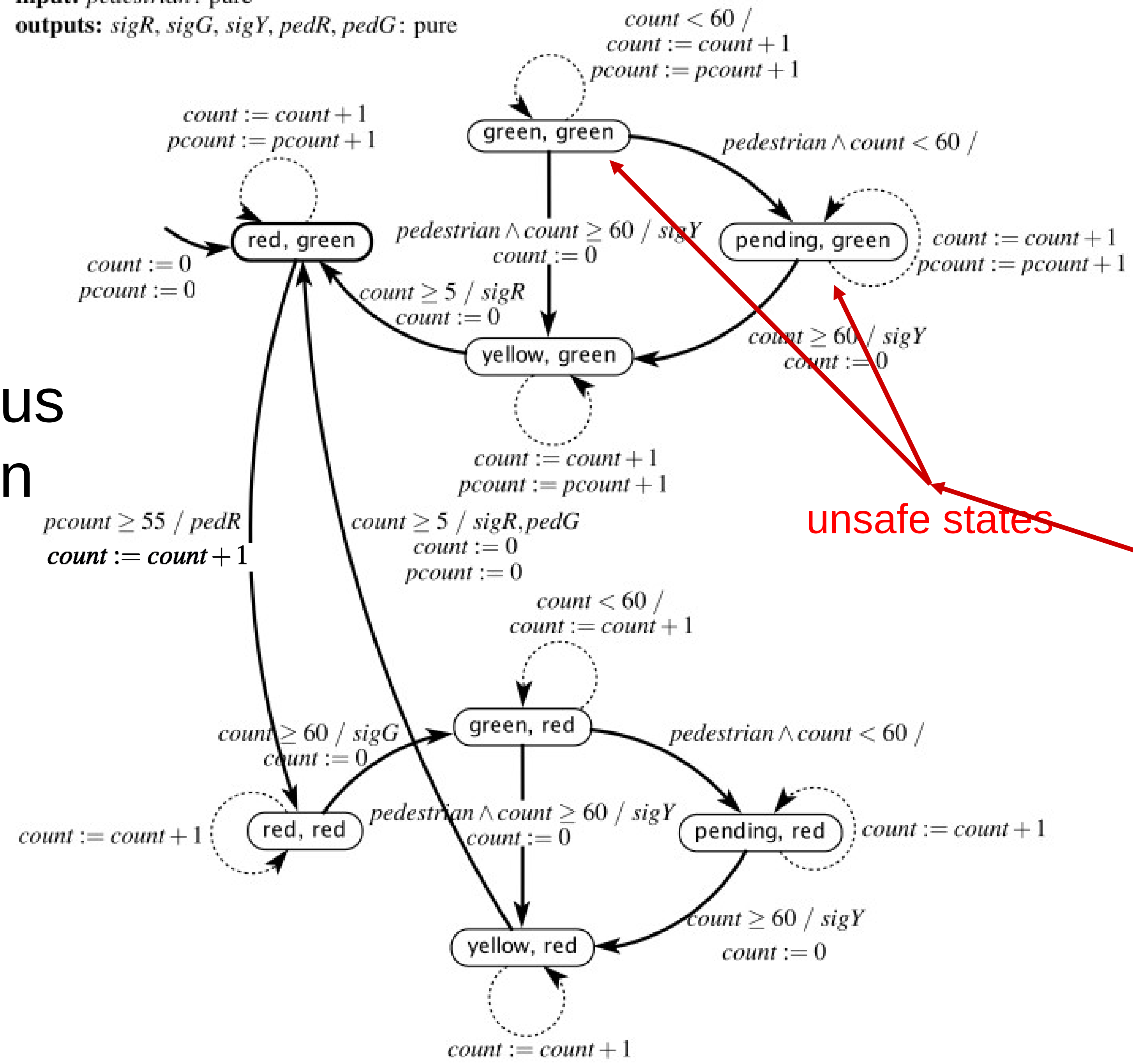




What is the size of the state space of the composite machine?

Synchronous composition

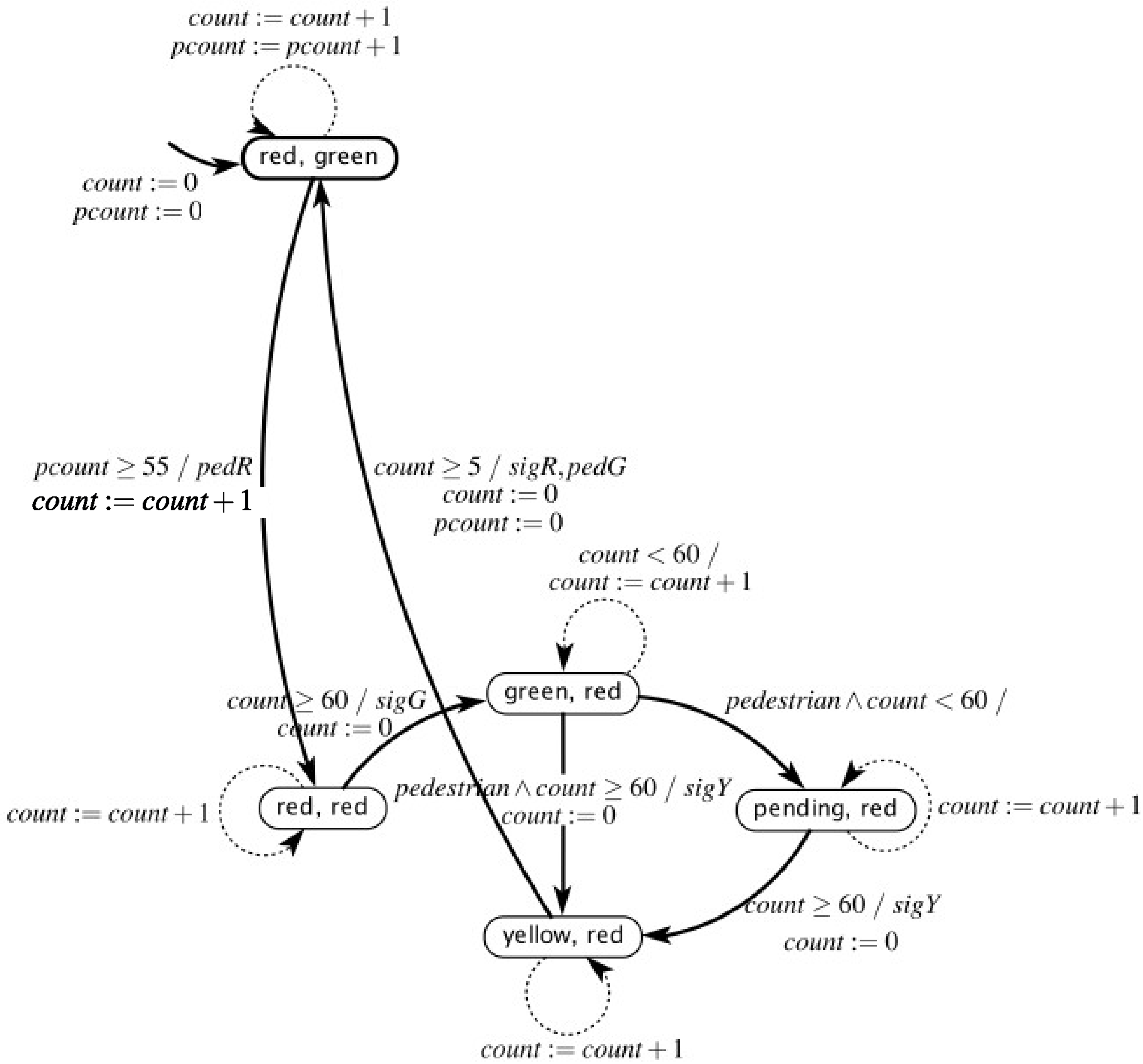
variables: $count$: $\{0, \cdots, 60\}$, $pcount$: $\{0, \cdots, 55\}$
input: $pedestrian$: pure
outputs: $sigR$, $sigG$, $sigY$, $pedR$, $pedG$: pure



unsafe states

Synchronous composition with unreachable states removed

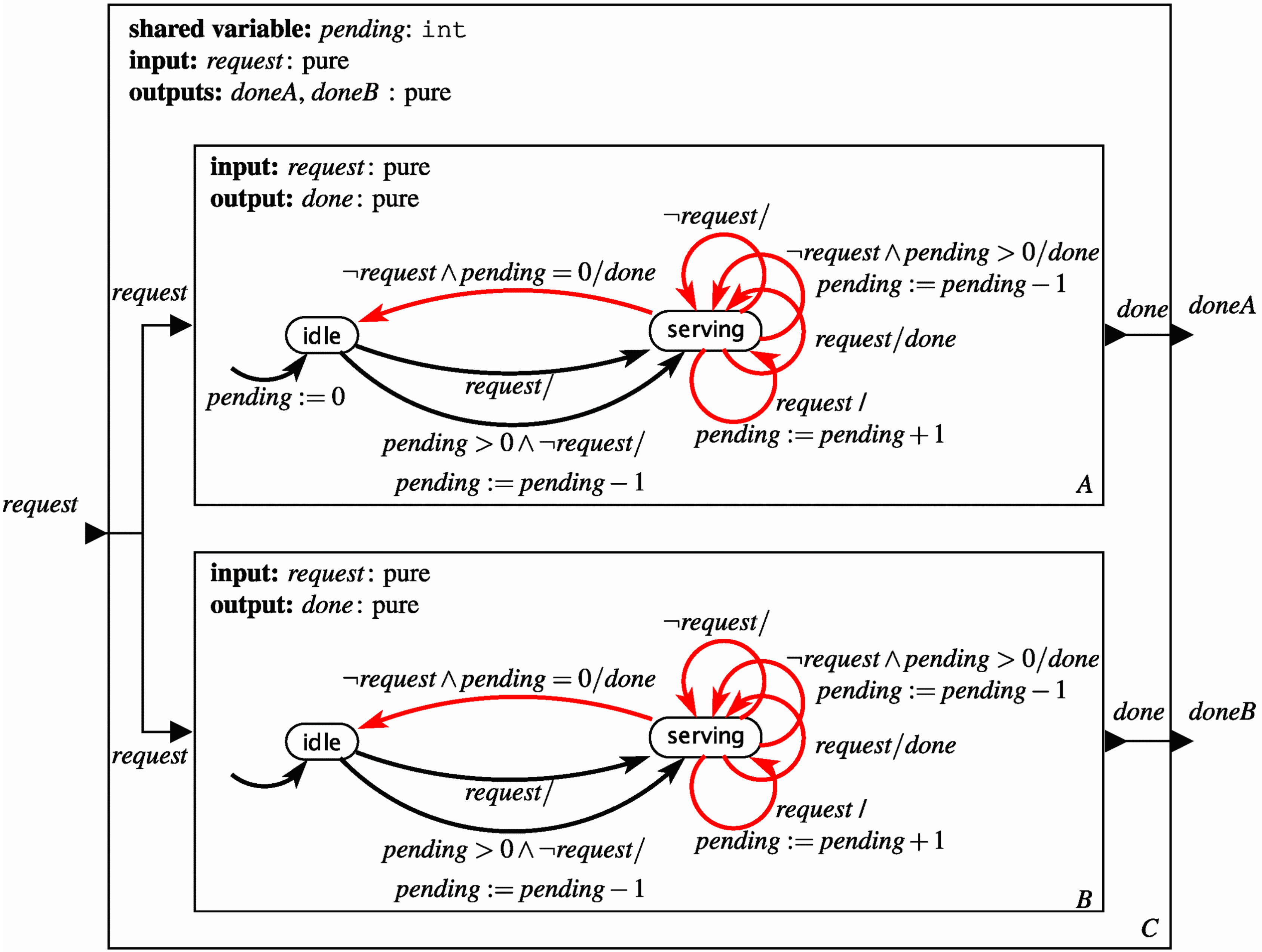
variables: $count$: $\{0, \cdots, 60\}$, $pcount$: $\{0, \cdots, 55\}$
input: $pedestrian$: pure
outputs: $sigR$, $sigG$, $sigY$, $pedR$, $pedG$: pure



Shared Variables

- Until now, the models were independent
- It is possible that the two models have shared variables
  - i.e. variables which can we written / read by both models
- Analysis much harder

Shared Variables: Two Servers



Shared Variables

- Potential problems in accessing the shared variable

- What happens if both models try to access (read or write) the variable at the same time?

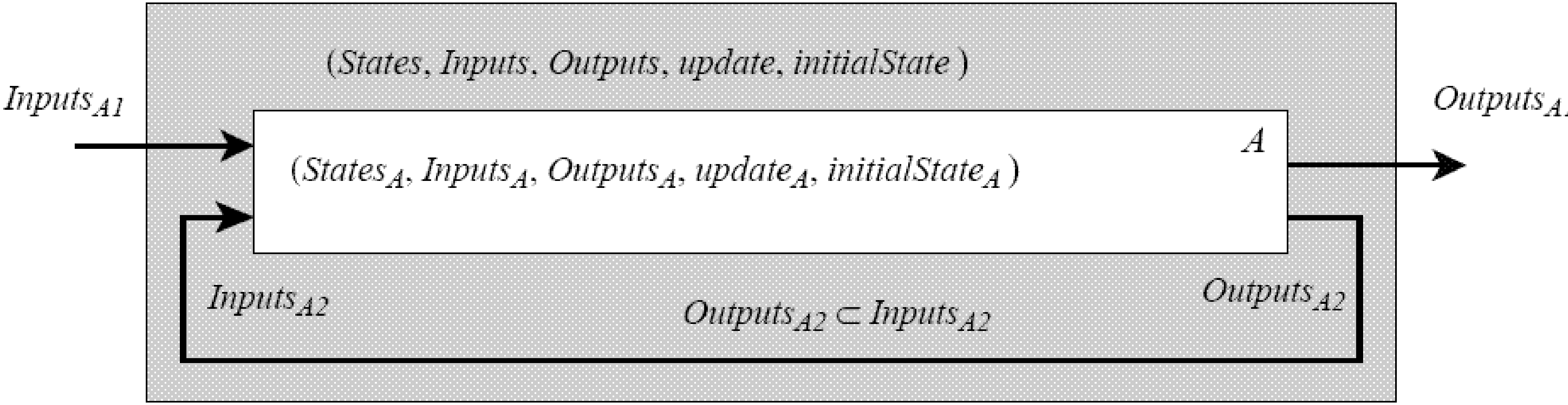
  - Answer: something bad. Might end up with an incorrect value
  - Solution: access to shared variable must be via atomic operations and guarded with a mutex

Shared Variables

- Atomic operation = an operation that is indivisible (once it starts, it can’t be interrupted until it ends)

- Mutex = a mechanism for ensuring only one process accesses a given resource (e.g. variable) at one time
  - A process first acquires the mutex, if it is available
  - Only afterwards it accesses the variable
  - While the mutex is acquired, no other process can access it
  - The process releases the mutex when it’s done with the variable

Feedback Composition


$(States, Inputs, Outputs, update, initialState)$
$Inputs_{A1}$
$Outputs_{A1}$
A
$(States_A, Inputs_A, Outputs_A, update_A, initialState_A)$
$Inputs_{A2}$
$Outputs_{A2}$
$Outputs_{A2} \subset Inputs_{A2}$

Reasoning about feedback composition can be very subtle.
(this topic is out of scope for EECS149/249A)