Composition of State Machines

01266212
CYBER PHYSICAL SYSTEM DESIGN

SEMESTER 1-2021

Original contents from
Edward A. Lee and Prabal Dutta, UC Berkeley, EECS 149/249A

Composition of State Machines

- State machines provide a convenient way to model behaviors of systems.
- One disadvantage that they have is that for most interesting systems, the number of states is very large, often even infinite.
- Any well-engineered system is a composition of simpler components.

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**: when do the components execute, relative to each other?

Temporal Composition of State Machines

Sequential vs. Concurrent

If Concurrent,

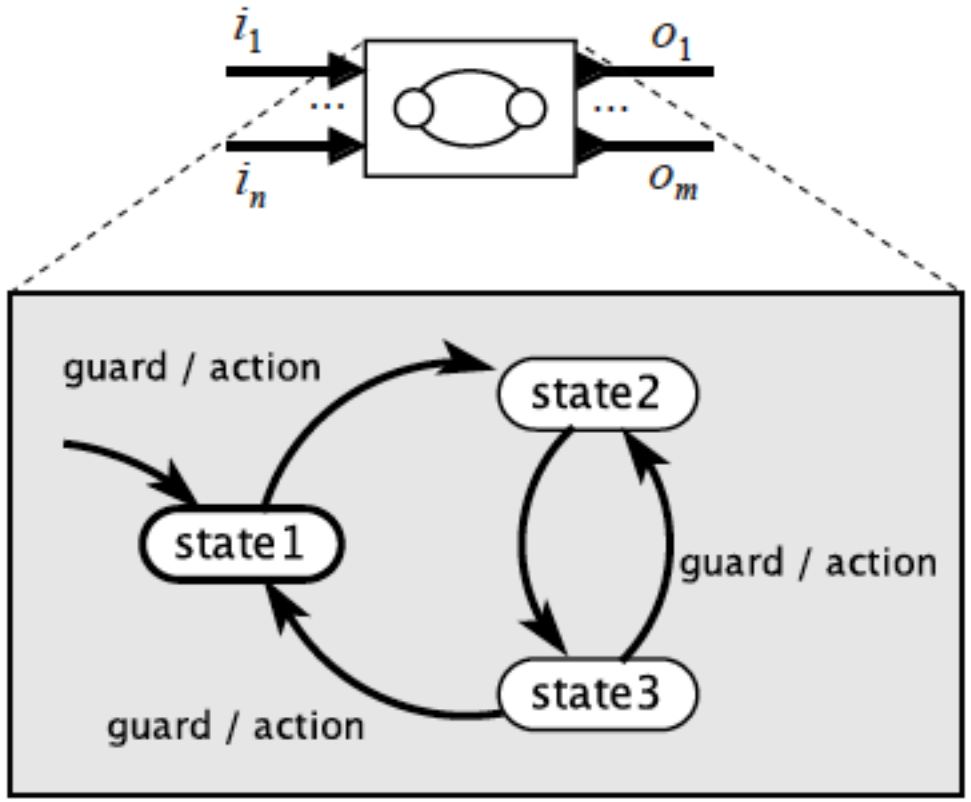
Asynchronous (the reactions from SMs are Independently)

vs.

Synchronous (the reactions from SMs are simultaneous.)

For concurrent composition, we need an interface. Actor Model for State Machines

Expose inputs and outputs, enabling concurrent composition:



Set-theoretic definition

- State machine is a 5-tuple:
  - (*S*, I, O, u, iniS), iniS $\in$ S
  - u : S x I → S x O
  - Where
    - S is a set of states
    - I is a set of input symbols
    - O is a set of symbols
    - u is an update fuinputnction
    - iniS is an initial state
- To apply the update function, if s $\in$ S and i $\in$ I:
  - (s',o) = u(s,i)

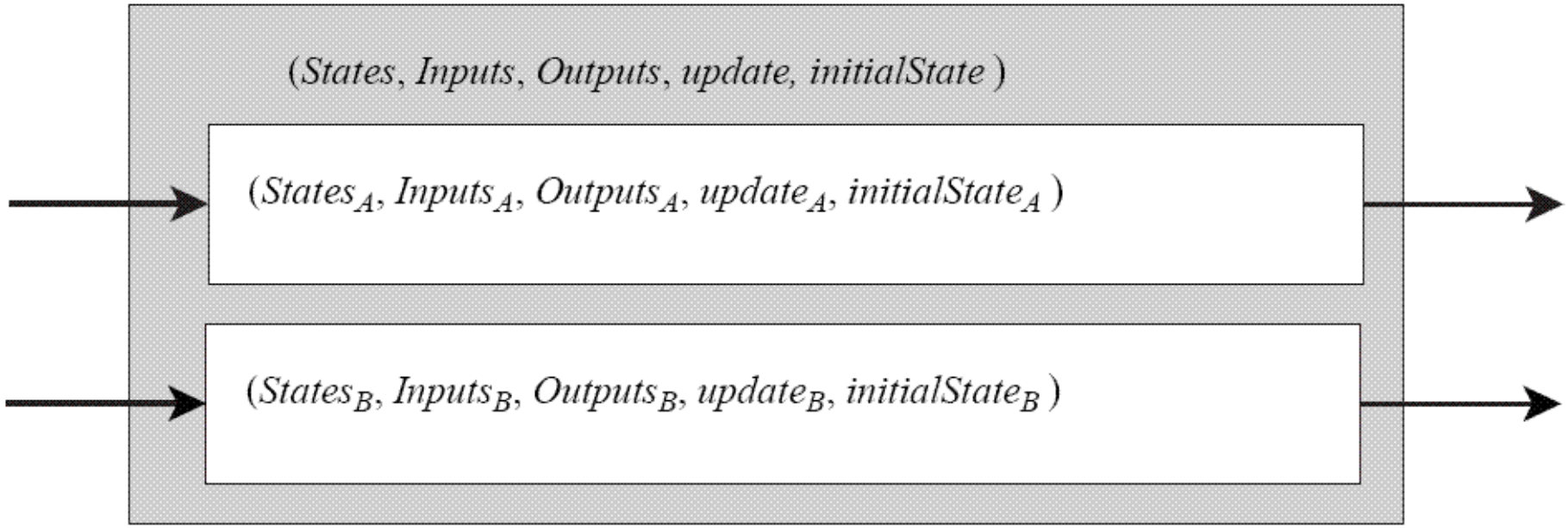
Spatial Composition of State Machines

Side-by-side composition

Cascade composition

Feedback composition

Side-by-Side Composition

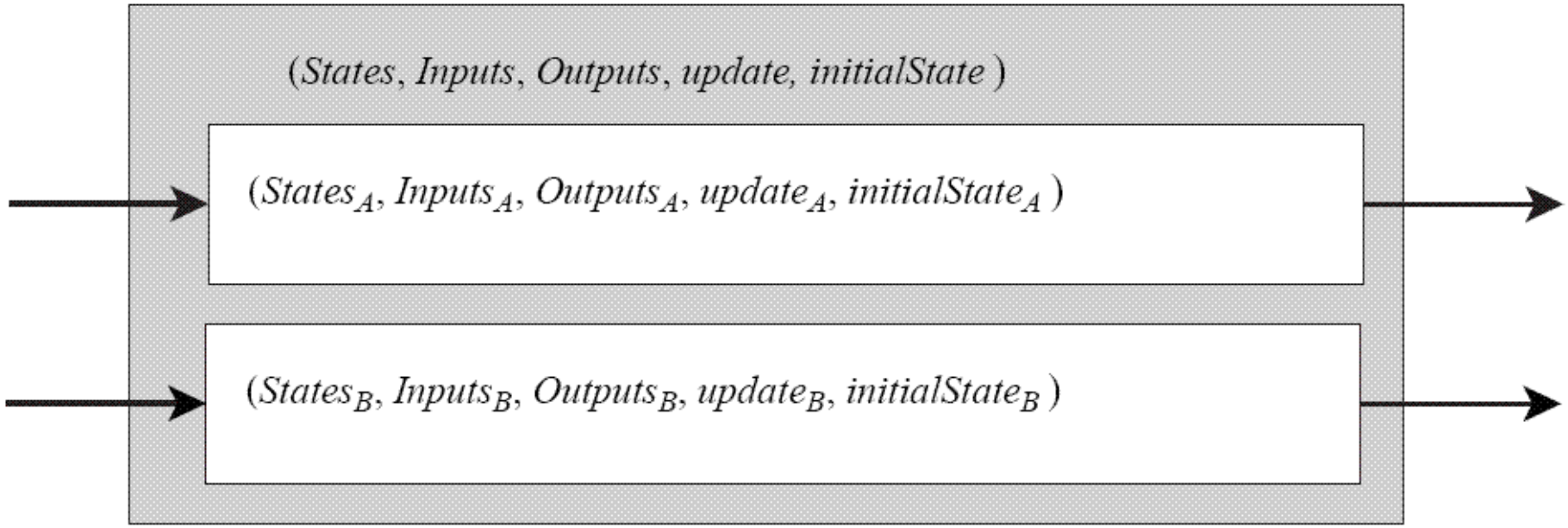


A key question: When do these machines react?

How the reactions of composed machines is coordinated is called a “Model of Computation” (MoC).

The state machines being composed do not communicate.

Side-by-Side, Parallel Composition



When do these machines react? Two of many possibilities:

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

Composition of State Machines

The state machines *A* and *B* are given by the five tuples,

$$\begin{aligned} A &= (States_A, Inputs_A, Outputs_A, update_A, initialState_A) \\ B &= (States_B, Inputs_B, Outputs_B, update_B, initialState_B) \end{aligned}$$

Then the synchronous side-by-side composition C is given by

$$\begin{aligned} States_C &= States_A \times States_B \\ Inputs_C &= Inputs_A \times Inputs_B \\ Outputs_C &= Outputs_A \times Outputs_B \\ initialState_C &= (initialState_A, initialState_B) \end{aligned}$$

$$update_C((s_A, s_B), (i_A, i_B)) = ((s'_A, s'_B), (o_A, o_B)),$$

where

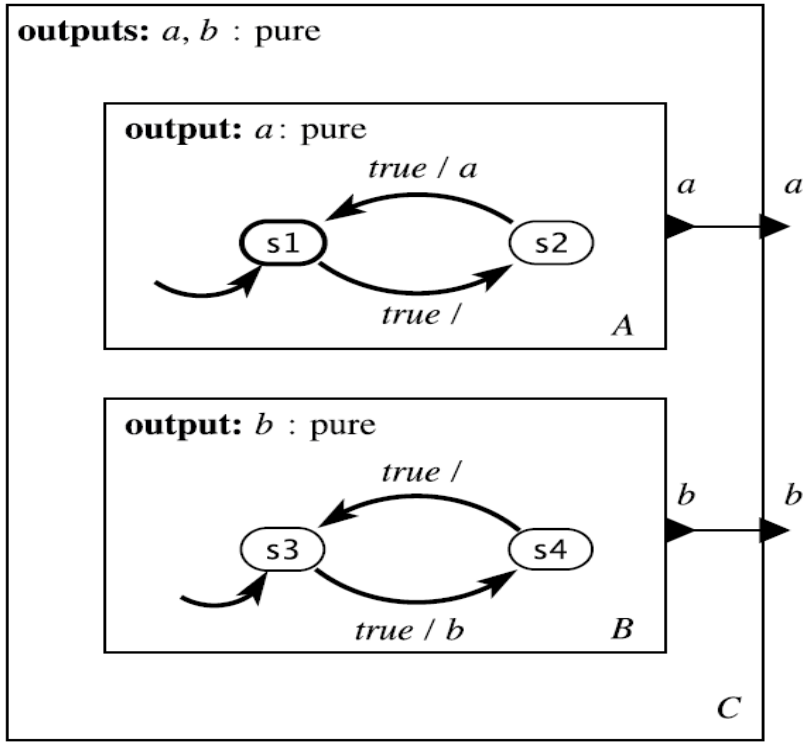
$$(s'_A, o_A) = update_A(s_A, i_A),$$

and

$$(s'_B, o_B) = update_B(s_B, i_B),$$

for all $s_A \in States_A$, $s_B \in States_B$, $i_A \in Inputs_A$, and $i_B \in Inputs_B$.

Side-by-side Composition of Two Actors



Consider FSMs *A* and *B* in Figure.

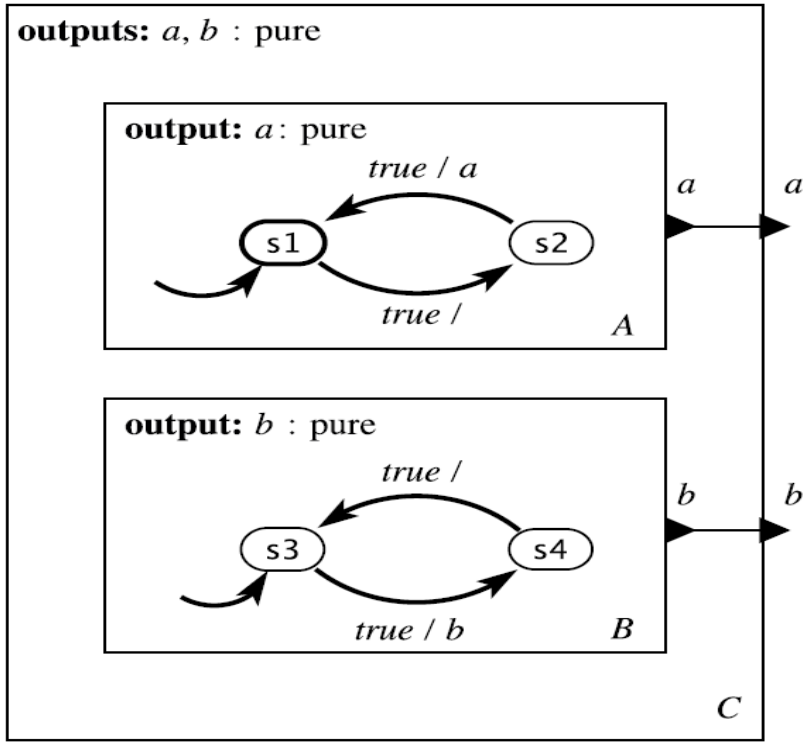
A has a single pure output *a*, and *B* has a single pure output *b*.

The side-by-side composition C has two pure outputs, *a* and *b*.

If the composition is synchronous, then on the first reaction, *a* will be absent, and *b* will be present. On the second reaction, it will be the reverse.

On subsequent reactions, *a* and *b* will continue to alternate being present.

Side-by-side Composition of Two Actors



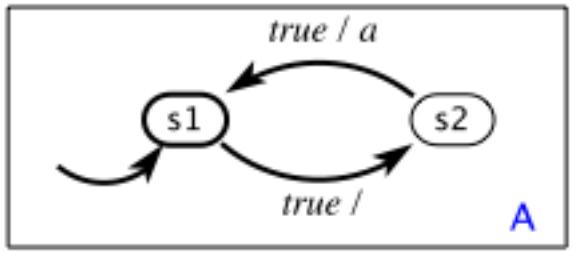
If the two state machines A and B are deterministic, then the synchronous side-by-side composition is also deterministic.

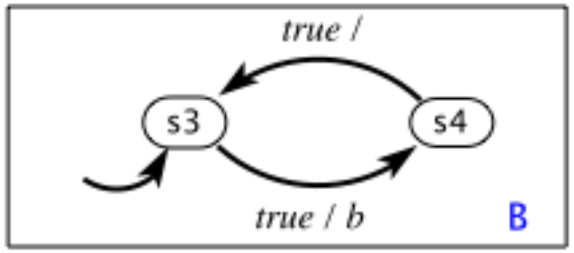
If a property held by the components is also a property of the composition, a property is **compositional**.

Synchronous Composition

$$S_C \subseteq S_A \times S_B$$

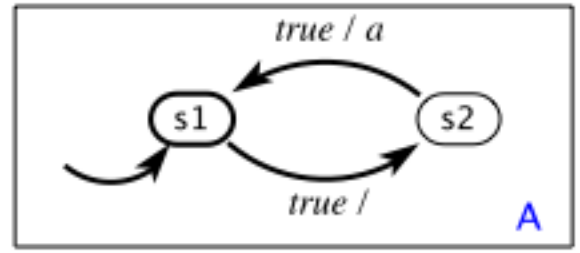
outputs: a, b (pure)

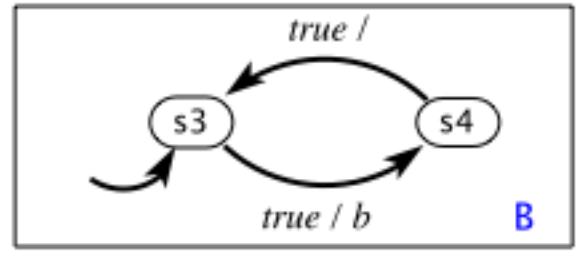


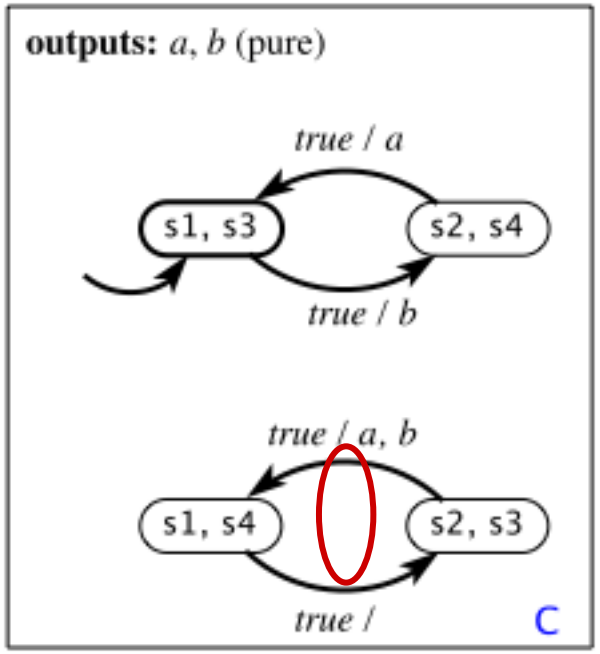


Synchronous Composition

$S_C \subseteq S_A \times S_B$







Synchronous composition

Note that these two states are not reachable.

Composition multiplies the state space

Asynchronous Composition

In an **asynchronous composition** of state machines, the component machines react independently.

It has several different interpretations, and each give a semantics to the composition (how to define a reaction of the composition)

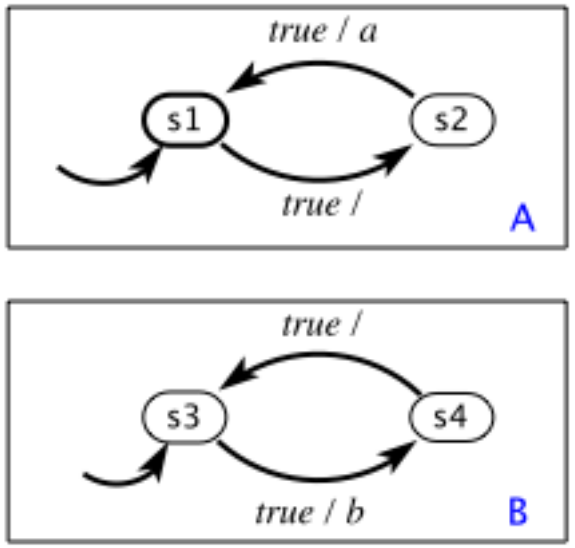
Semantics 1. A reaction of C is a reaction of one of A or B, where the choice is **nondeterministic**. (an **interleaving semantics)**

Semantics 2. A reaction of C is a reaction of A, B, or both A and B, where the choice is **nondeterministic**. A variant of this possibility might allow neither to react.

Asynchronous Composition

$$S_C \subseteq S_A \times S_B$$

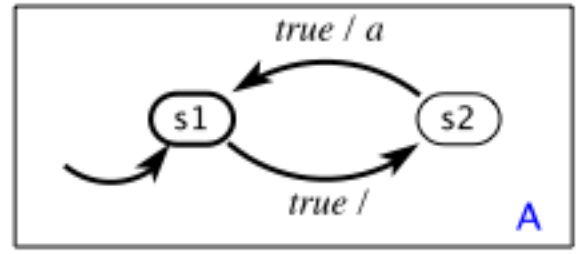
outputs: *a*, *b* (pure)

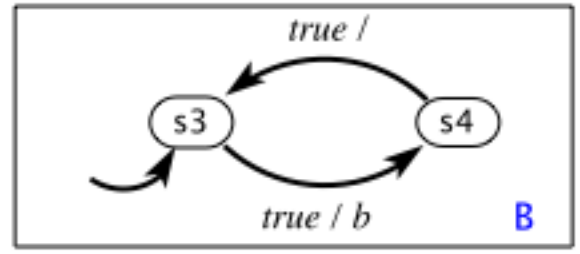


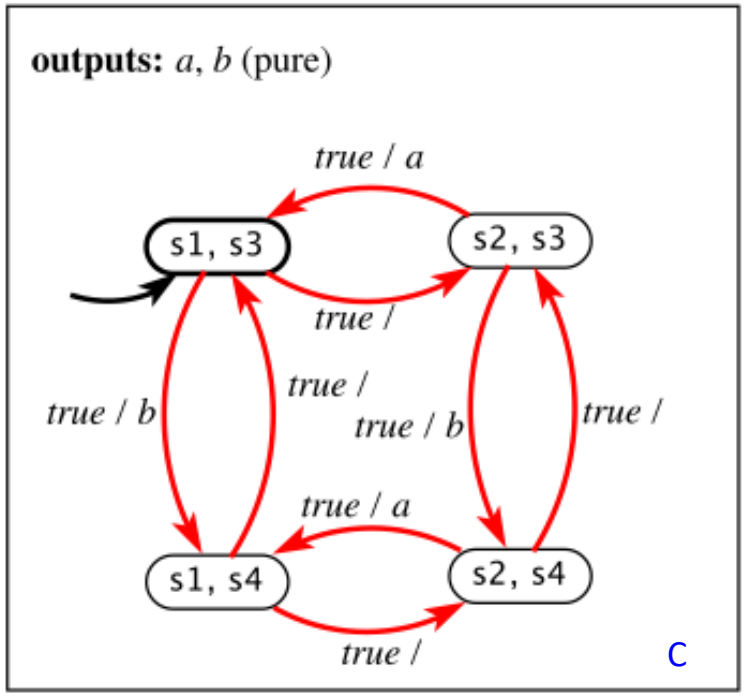
Asynchronous composition using interleaving semantics

Asynchronous Composition (is not synchronous composition with stuttering transitions)

$$S_C \subseteq S_A \times S_B$$







Asynchronous composition using interleaving semantics

Note that now all states are reachable.

Composition of State Machines

The state machines *A* and *B* are given by the five tuples,

$$
\begin{aligned}
A &= (States_A, Inputs_A, Outputs_A, update_A, initialState_A) \\
B &= (States_B, Inputs_B, Outputs_B, update_B, initialState_B)
\end{aligned}
$$

Then the synchronous side-by-side composition C is given by

$$
\begin{aligned}
States_C &= States_A \times States_B \\
Inputs_C &= Inputs_A \times Inputs_B \\
Outputs_C &= Outputs_A \times Outputs_B \\
initialState_C &= (initialState_A, initialState_B)
\end{aligned}
$$

$$update_C((s_A, s_B), (i_A, i_B)) = ((s'_A, s'_B), (o'_A, o'_B)),$$

where either

$$(s'_A, o'_A) = update_A(s_A, i_A) \text{ and } s'_B = s_B \text{ and } o'_B = absent$$

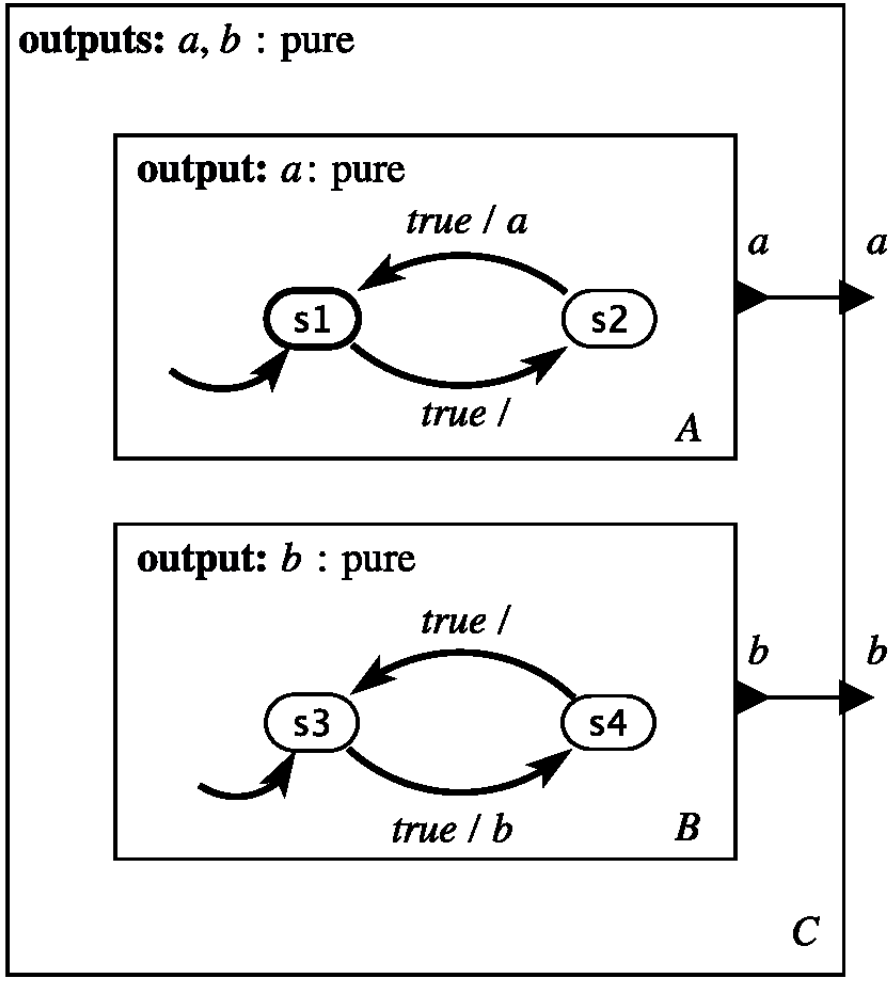
or

$$(s'_B, o'_B) = update_B(s_B, i_B) \text{ and } s'_A = s_A \text{ and } o'_A = absent$$

for all $s_A \in States_A$, $s_B \in States_B$, $i_A \in Inputs_A$, and $i_B \in Inputs_B$.

Syntax vs. Semantics

The answers to these questions defines the MoC being used.



Synchronous or Asynchronous composition?

If asynchronous, does it allow simultaneous transitions in A & B? How to choose whether A or B reacts when C reacts?

Scheduling Semantics for Asynchronous Composition

In previous cases, the choice of which component machine reacts is nondeterministic.

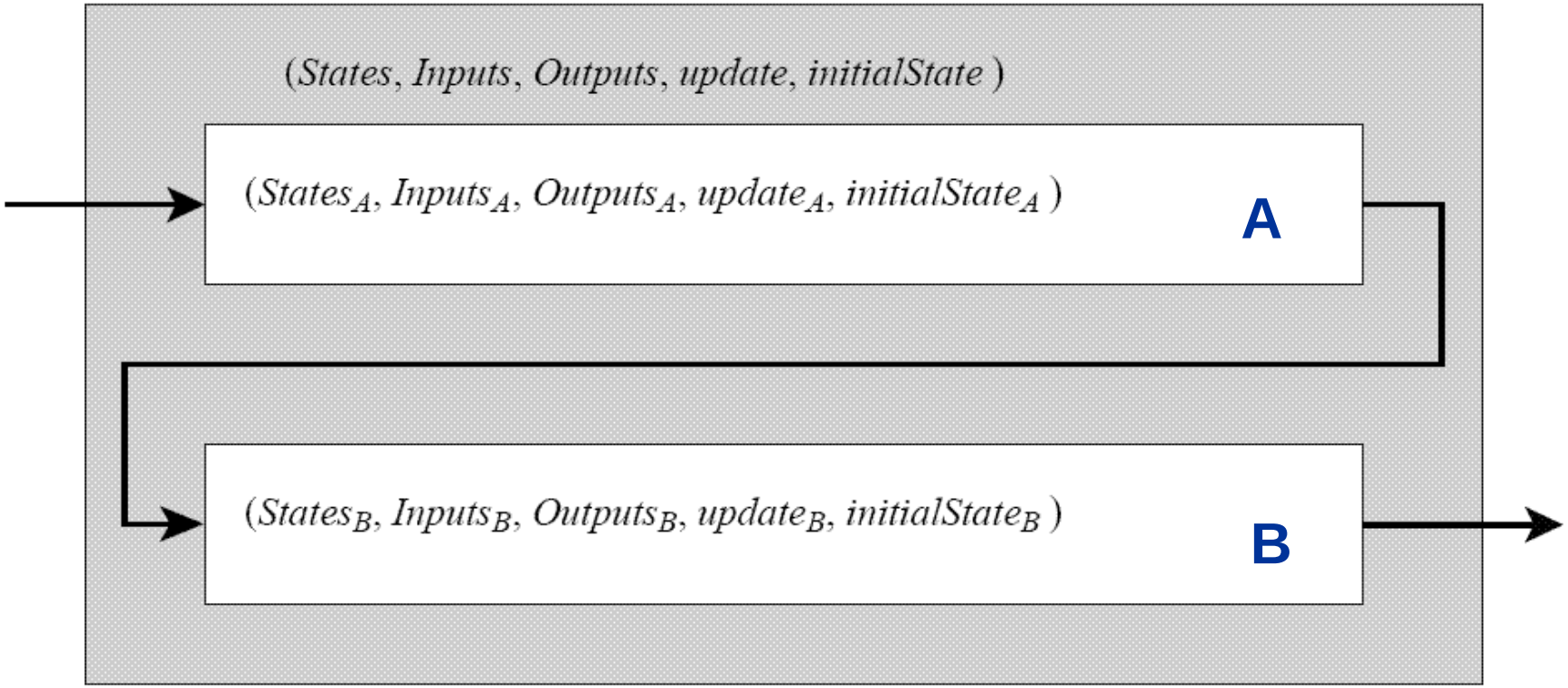
It is often more useful to introduce some scheduling policies, where the environment is able to influence or control the nondeterministic choice.

Semantics 3. A reaction of *C* is a reaction of one of *A* or *B*, where the environment chooses which of *A* or *B* reacts.

Semantics 4. A reaction of *C* is a reaction of *A*, *B*, or both *A* and *B*, where the choice is made by the environment.

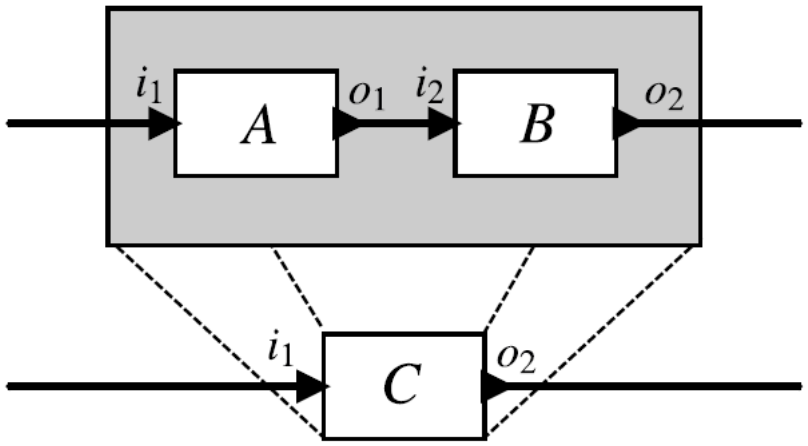
To implement semantics 3 and 4, a composition has to provide some mechanism for the environment to choose which component machine should react (for scheduling the component machines).

Cascade Composition



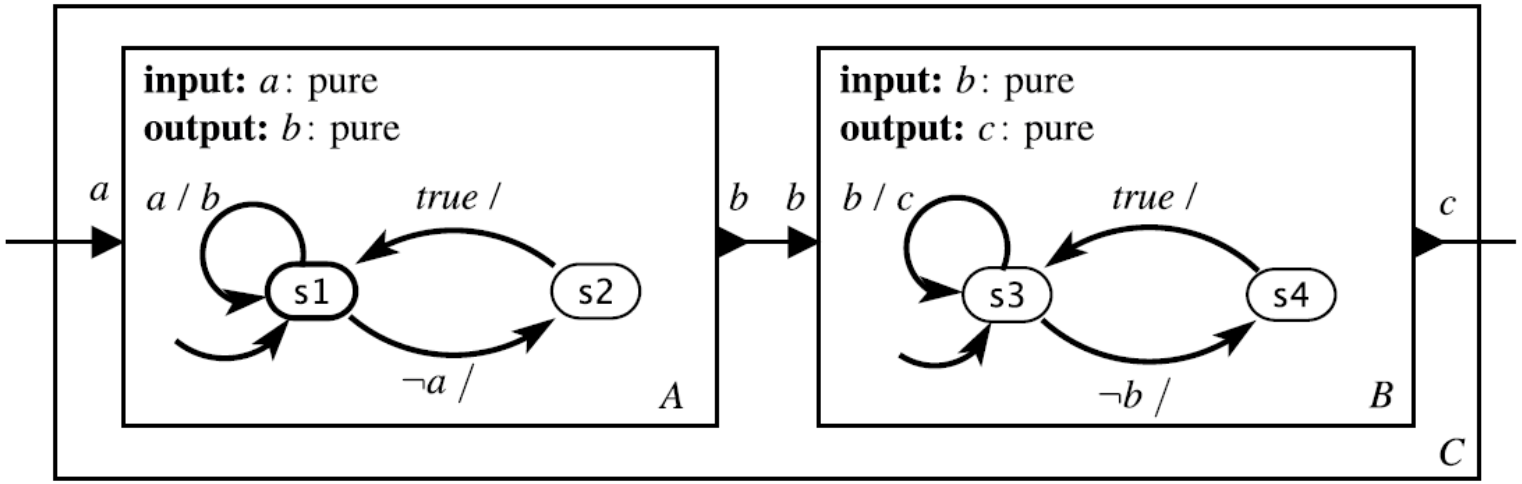
Output port(s) of A connected to input port(s) of B

Cascade Composition

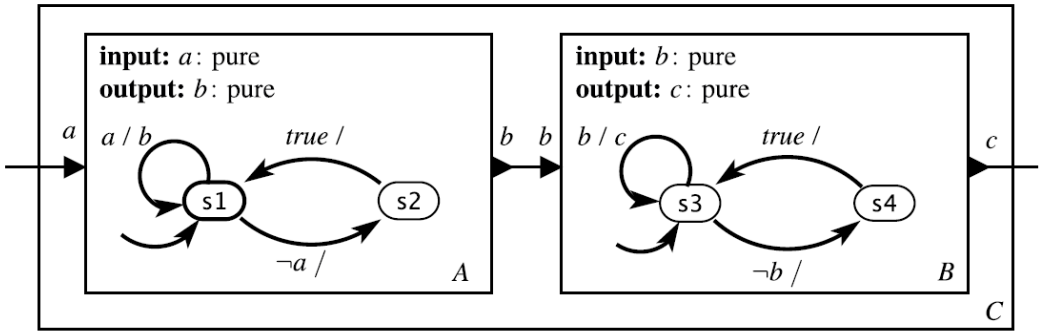


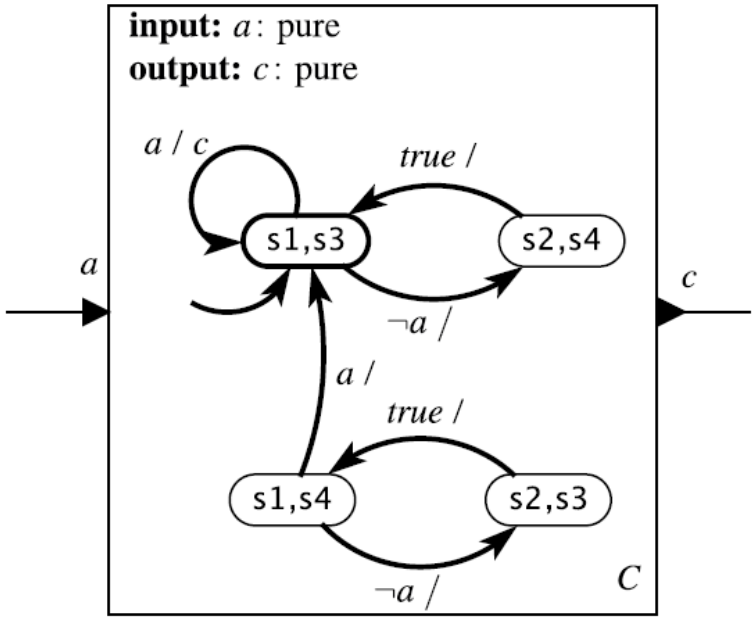
A reaction of *C* consists of a reaction of both *A* and *B*, where A reacts first, produces its output (if any), and then *B* reacts. Logically, we view this as occurring in zero time, so the two reactions are in a sense simultaneous and instantaneous. But they are causally related in that the outputs of *A* can affect the behavior of *B*.

Example of a cascade composition of two FSMs



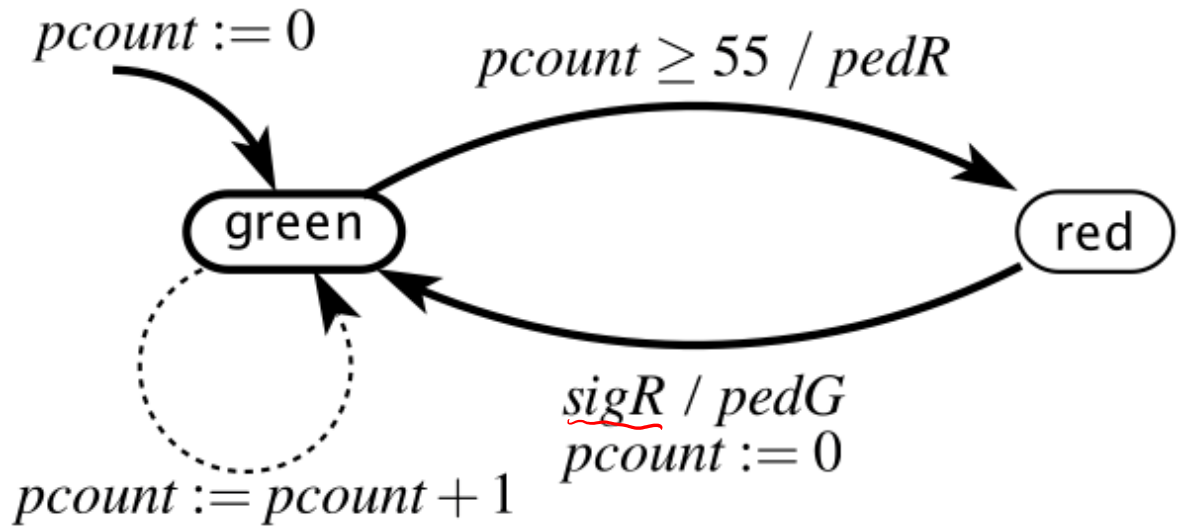
Cascade Composition Synchronous Composition





Example: Time-Triggered Pedestrian Light

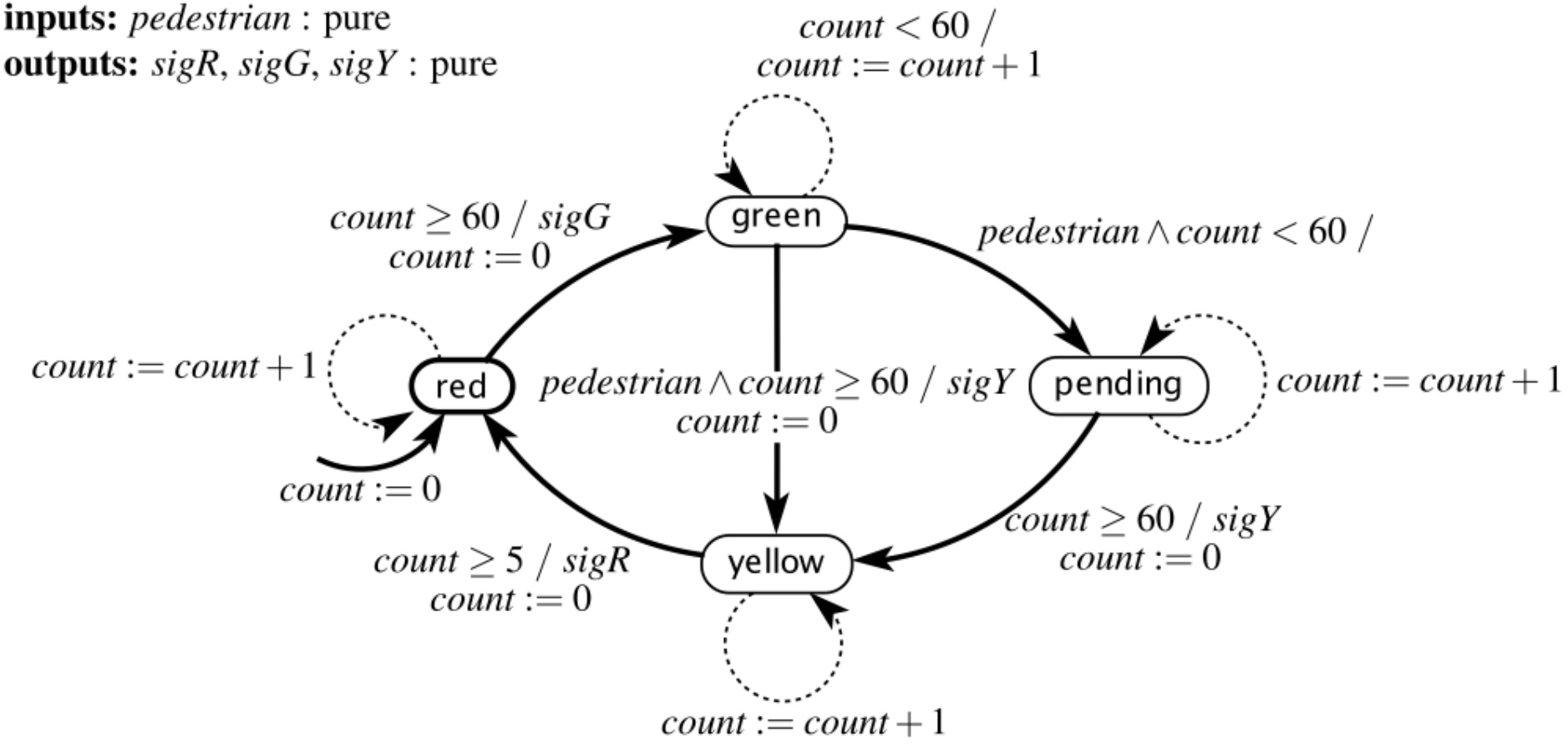
variable: $pcount$: $\{0, \cdots, 55\}$
input: $sigR$: pure
outputs: $pedG, pedR$: pure



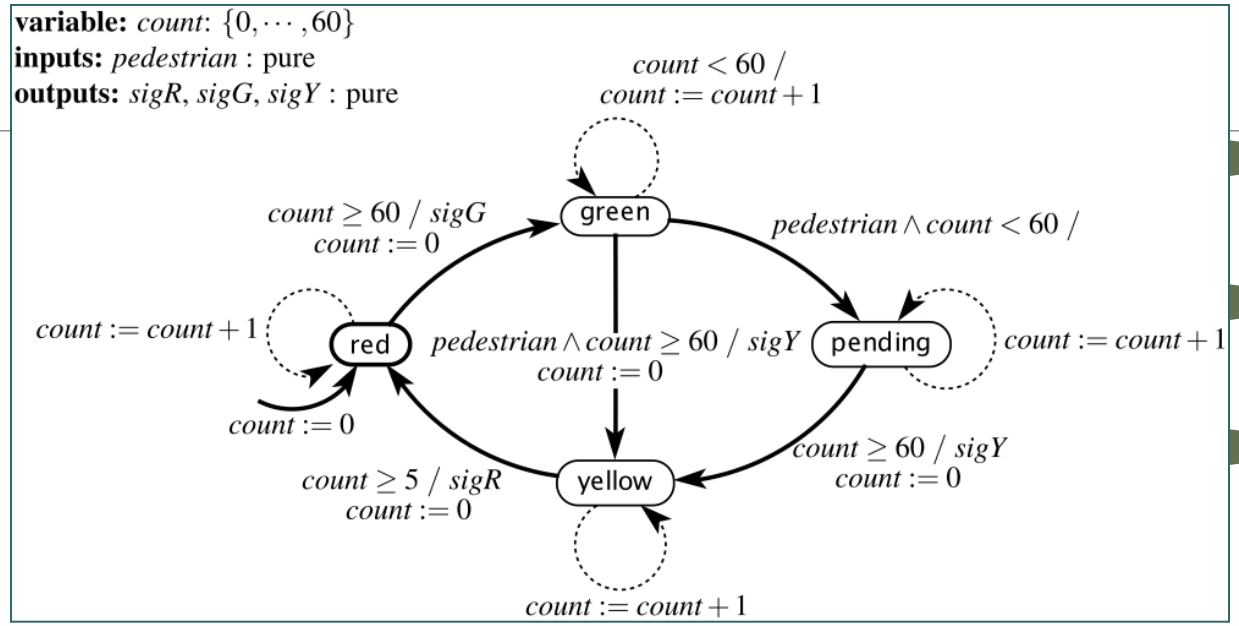
This light stays green for 55 seconds, then goes red.
Upon receiving a sigR input, it repeats the cycle.

Example: Time-Triggered Car Light

variable: *count*: $\{0, \cdots, 60\}$
inputs: *pedestrian* : pure
outputs: *sigR*, *sigG*, *sigY* : pure



Pedestrian Light with Car Light

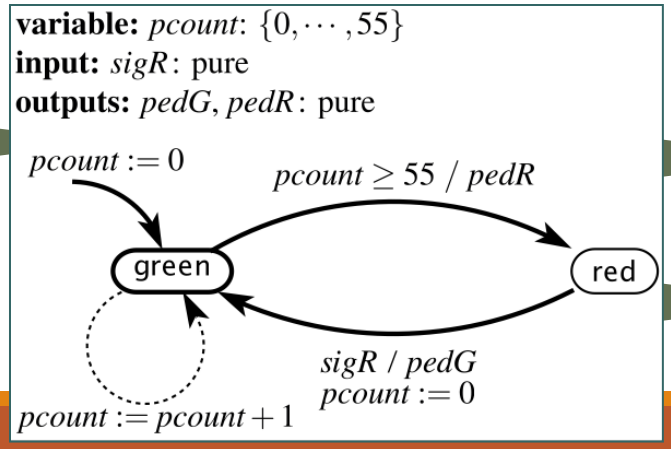


sigY

sigG

sigR

sigR

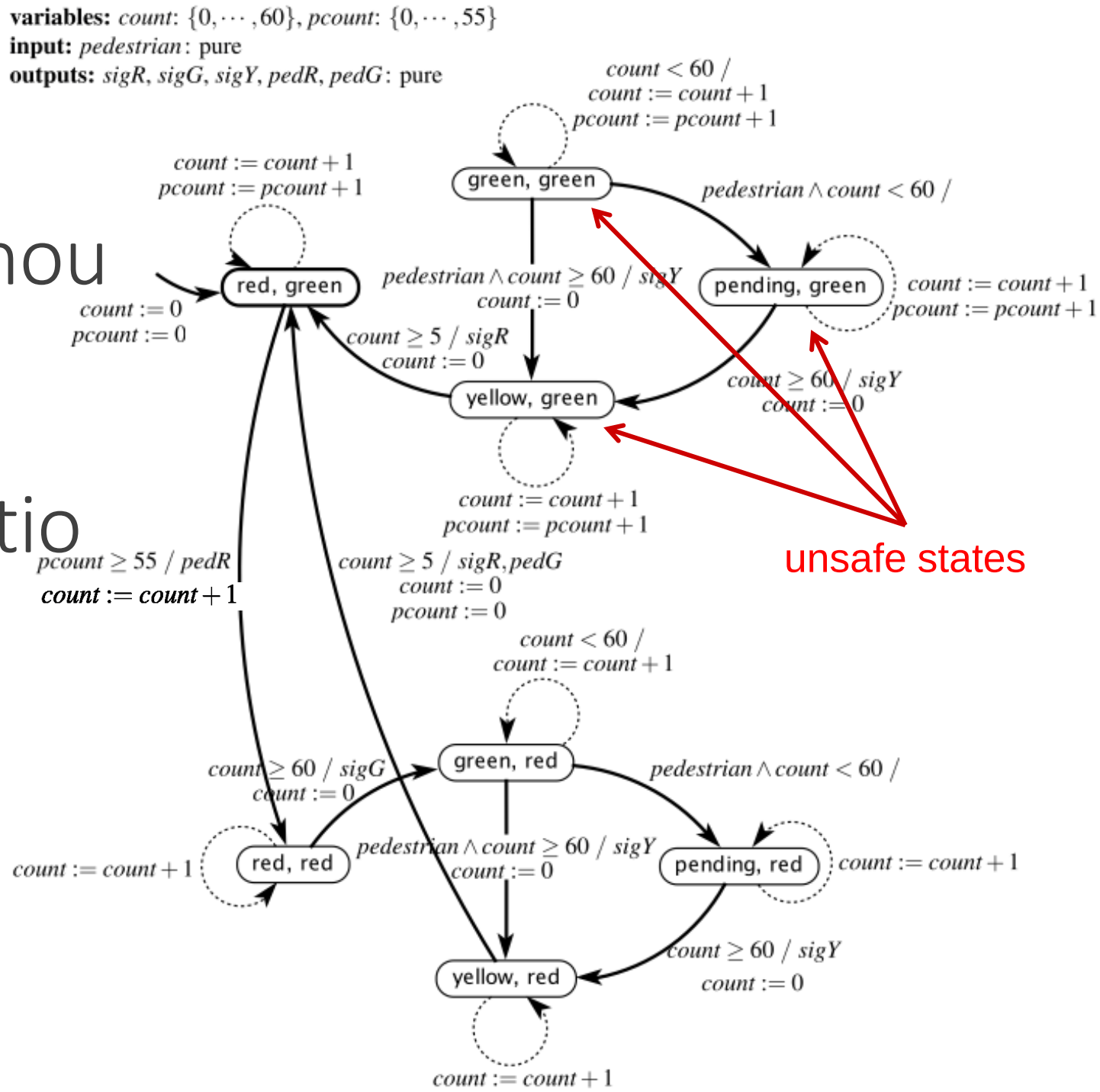


pedG

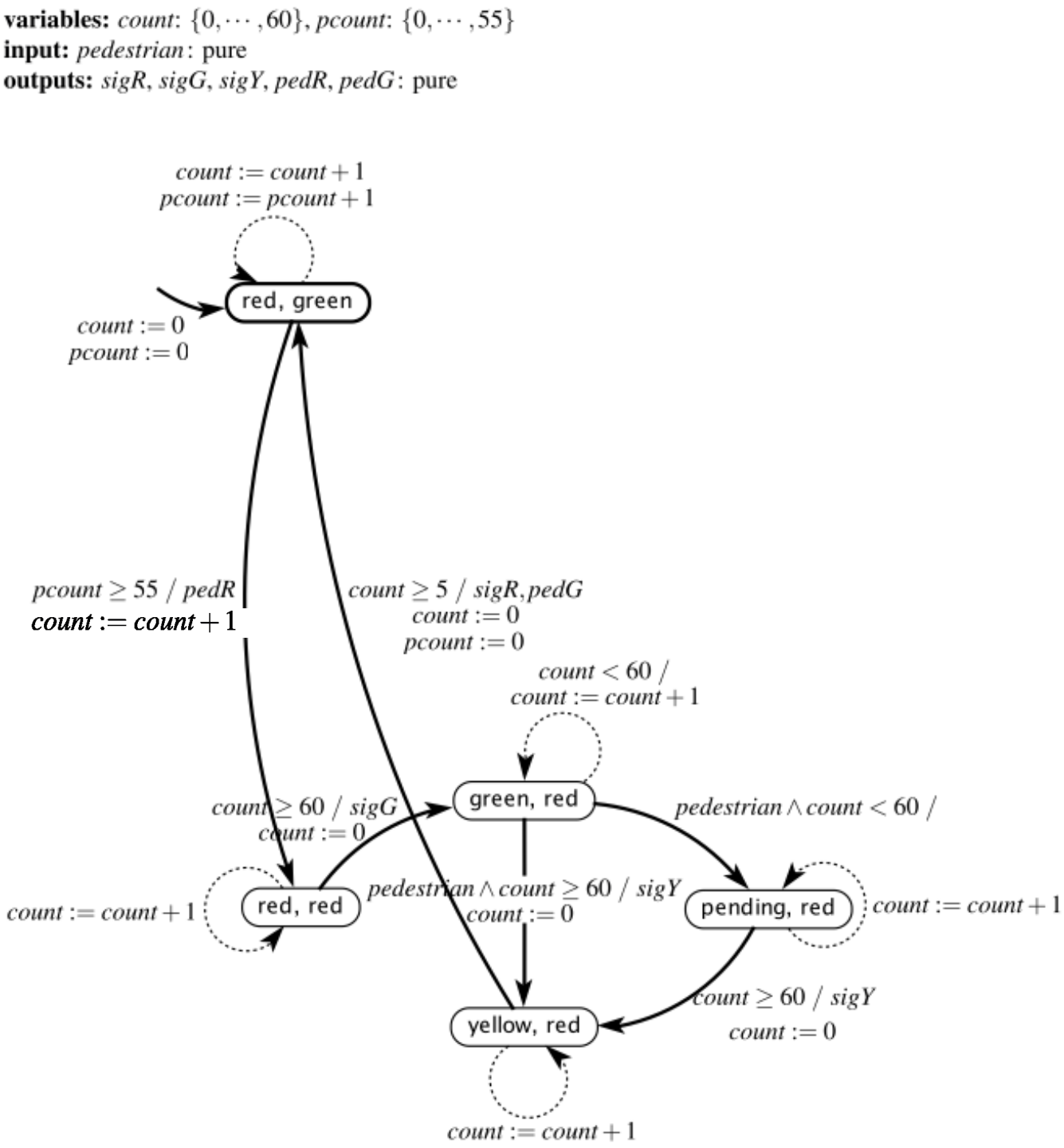
pedR

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

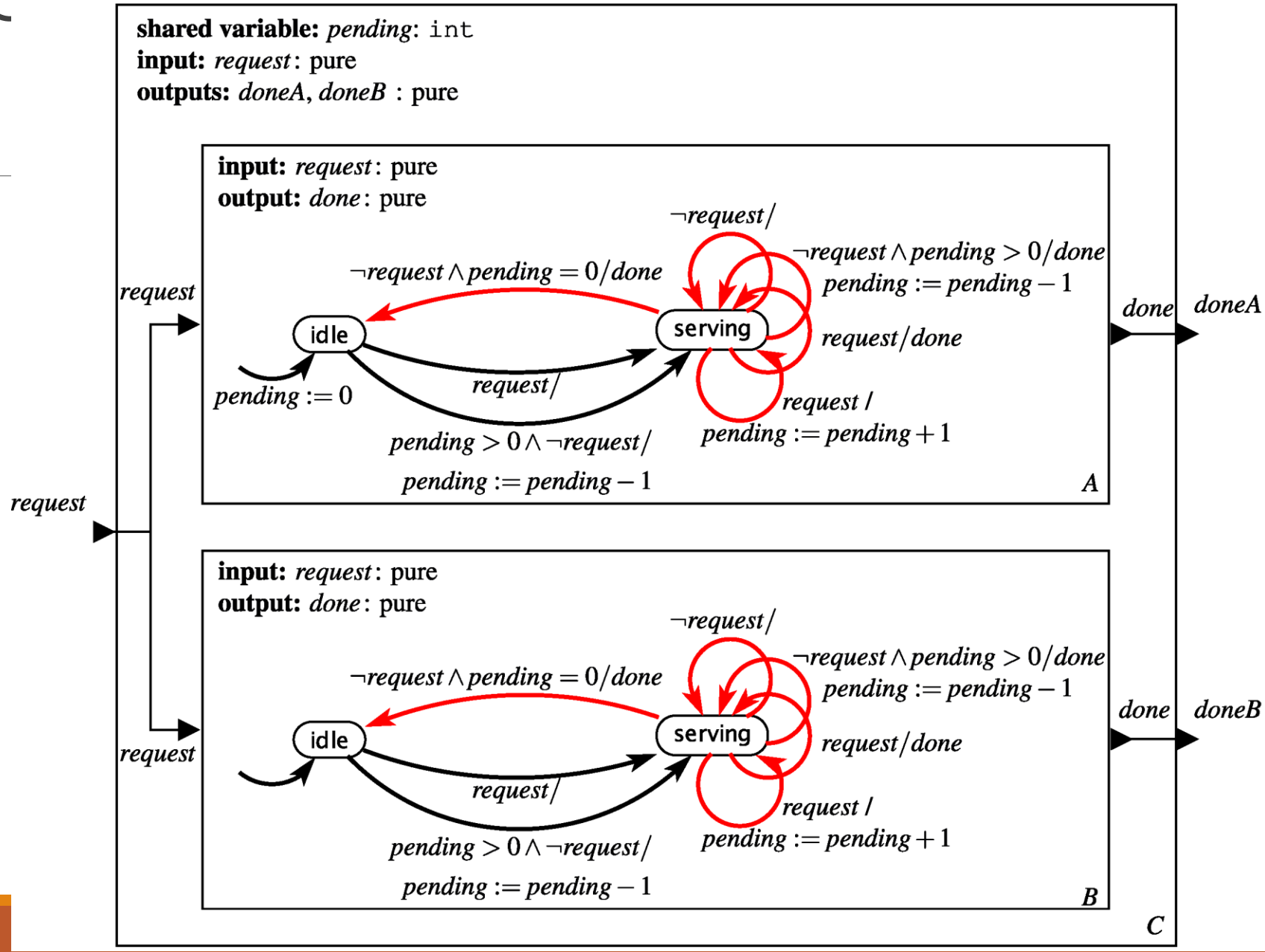
Synchronous cascade composition



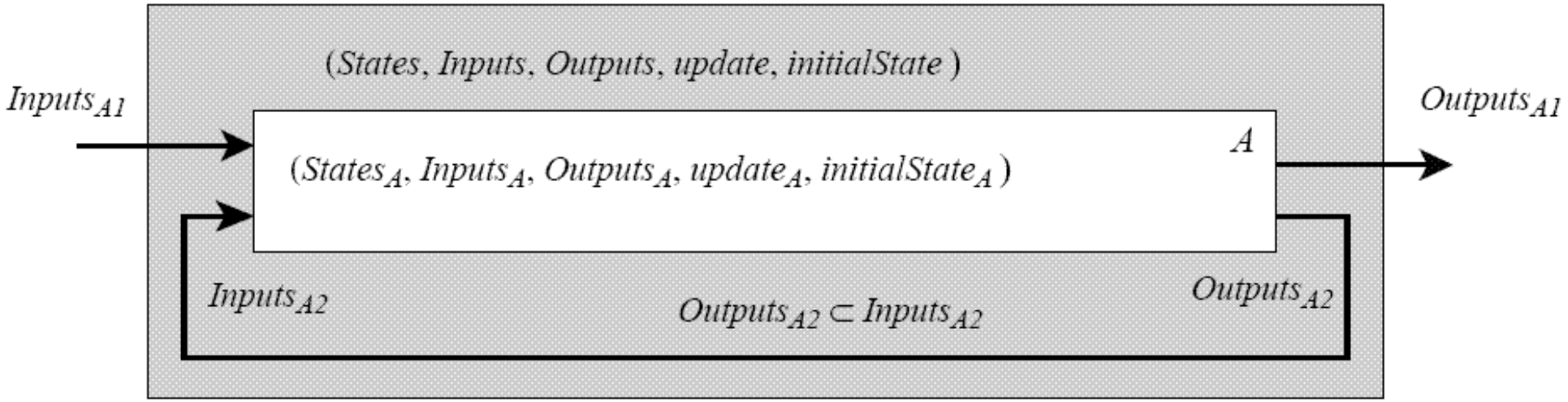
Synchronous composition with unreachable states removed



Shared Variables: Two Servers



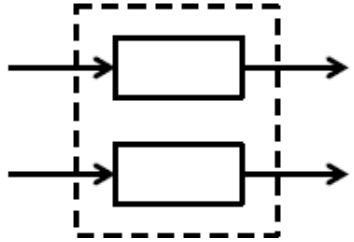
Feedback Composition



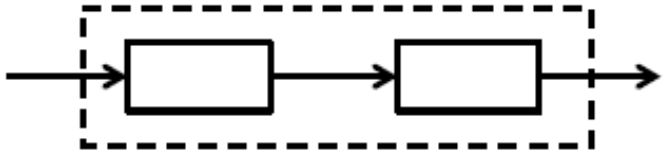
Reasoning about feedback composition can be very subtle.
(more about this later)

Spatial Composition of State Machines

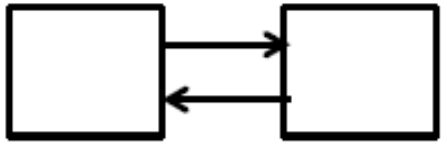
Side-by-side composition

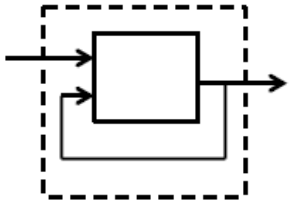


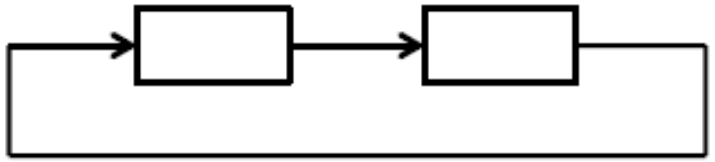
Cascade composition



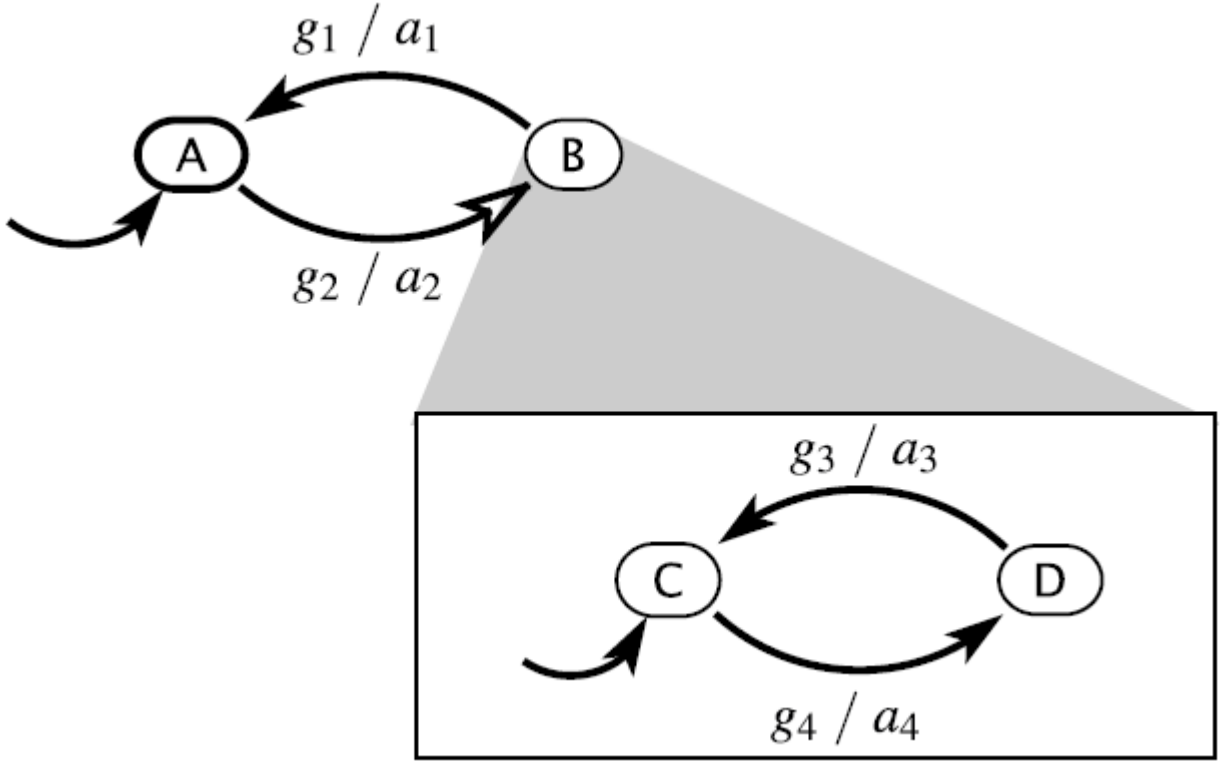
Feedback compositior







Hierarchical State Machines



A state may have a refinement that is another state machine.

Example, state B has a refinement that is another FSM with two states, C and D. What it means for the machine to be in state B is that it is in one of states C or D.

Reference

- Lee, Edward & Seshia, Sanjit. (2011). Introduction to Embedded Systems - A Cyber-Physical Systems Approach.
- Lecture Note Slides from EECS 149/249A: Introduction to Embedded Systems (UC Berkeley) by Prof. Prabal Dutta and Sanjit A. Seshia