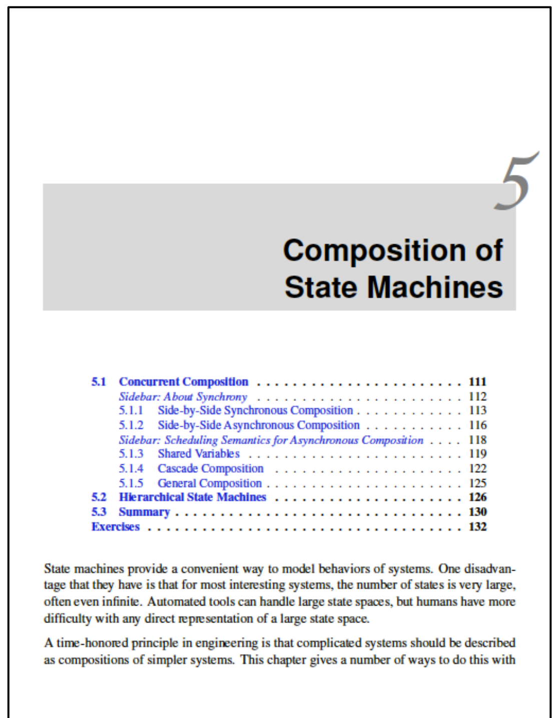
Lecture 11: Concurrent Composition

Slides adapted from Edward A. Lee

Outline

- Temporal composition
  - Sequential
  - Concurrent
- Spatial composition
  - Side-by-side composition
  - Cascade composition
  - Feedback composition

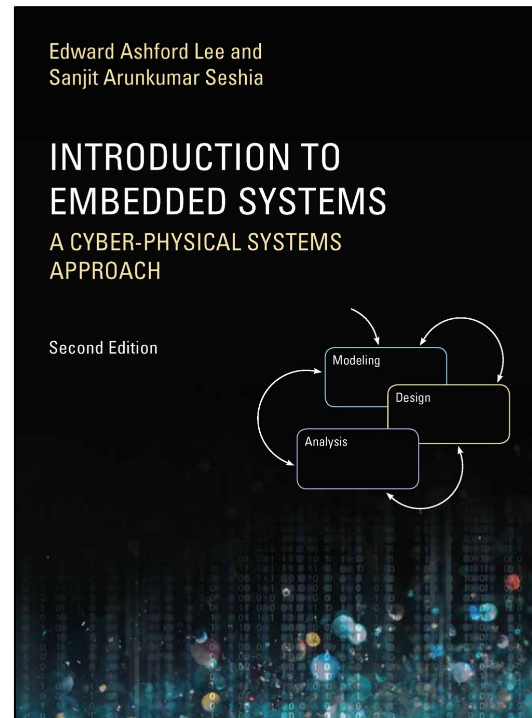


5

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. Automated tools can handle large state spaces, but humans have more difficulty with any direct representation of a large state space.

A time-honored principle in engineering is that complicated systems should be described as compositions of simpler systems. This chapter gives a number of ways to do this with



Composition of State Machines

- How do we construct complex state machines out of simpler "building blocks"?

- Two kinds of composition:

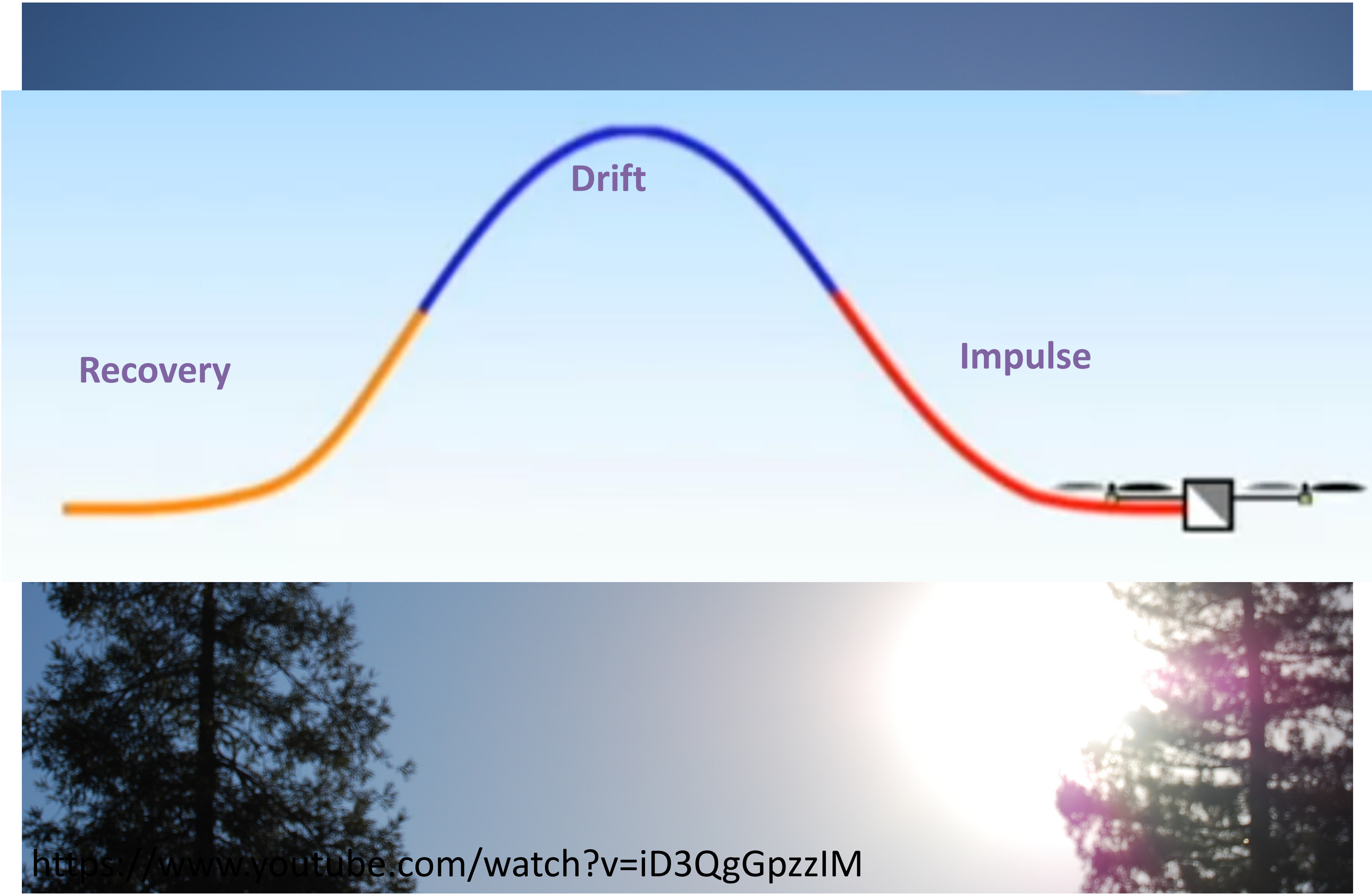
1. **Spatial**: how do the components communicate between each other?
2. **Temporal**: when do the components execute, relative to each other?

Temporal Composition of State Machines

- Sequential vs. Concurrent

- If Concurrent, Asynchronous vs. Synchronous

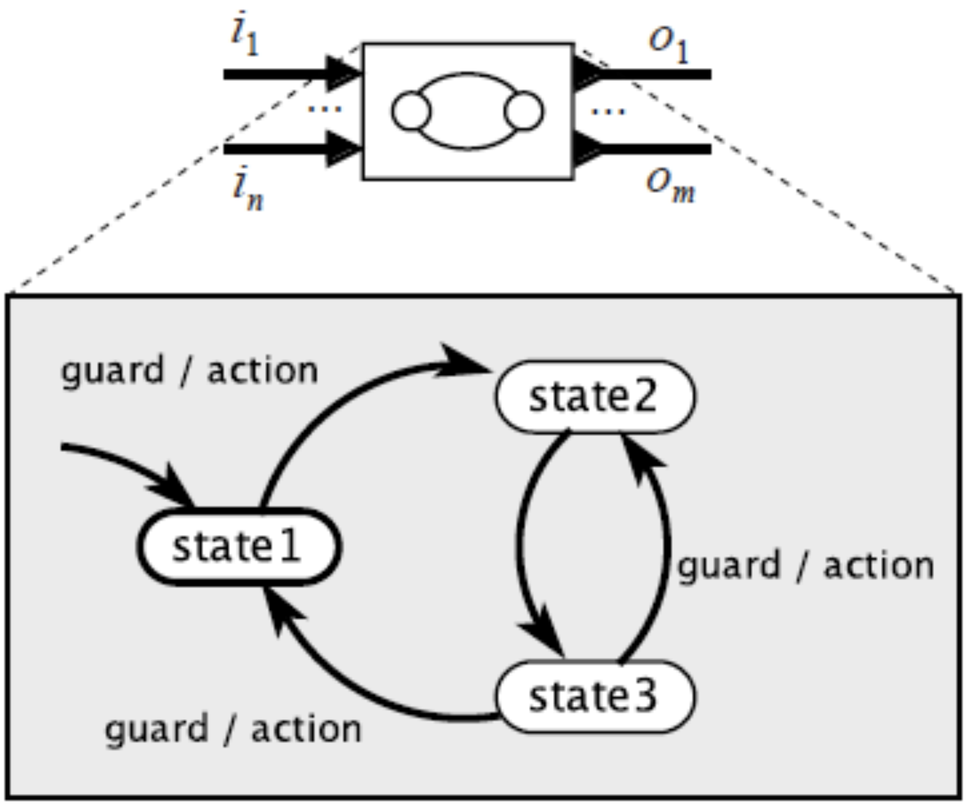
Hybrid Systems Provide *Sequential* Composition
Modal models: Sequencing between modes



[Tomlin et al.]

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:

  $$(S, I, O, u, i), \; i \in S$$

  $$u : S \times I \to S \times O$$

  Where

  - S is a set of states
  - I is a set of input symbols
  - O is a set of output symbols
  - u is an update function
  - i is an initial state

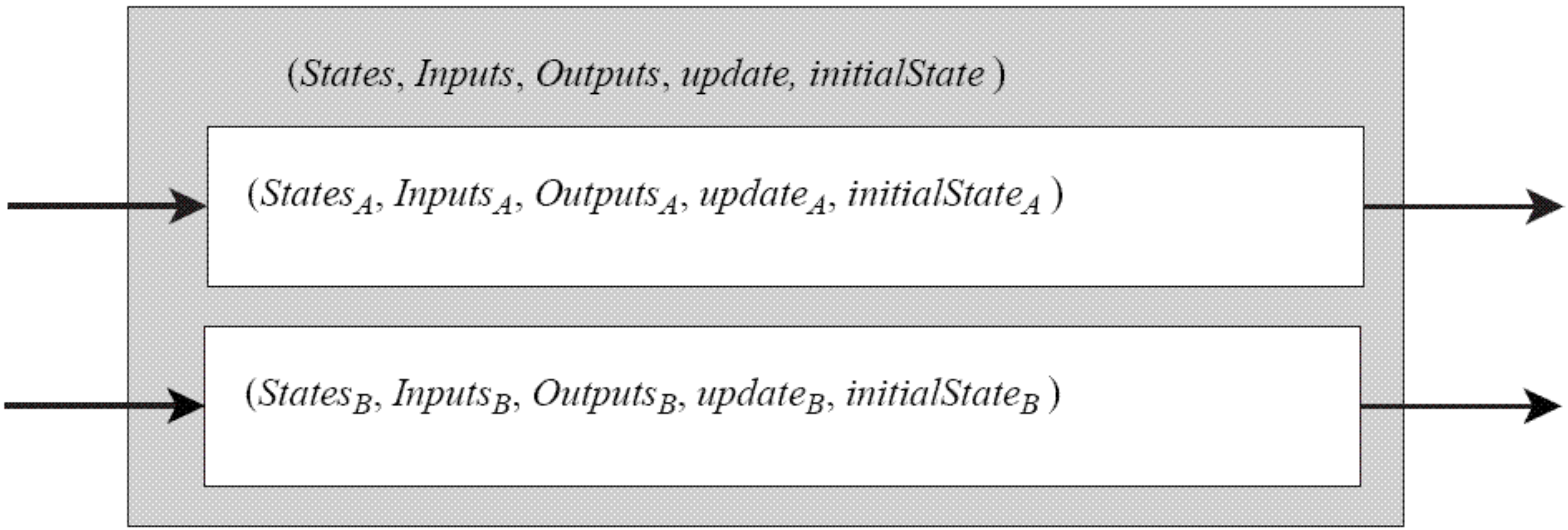
- To apply the update function, if $s \in S$ and $c \in I$

  $$(s', o) = u(s, c)$$

Spatial Composition of State Machines

- Side-by-side composition
- Cascade composition
- Feedback composition

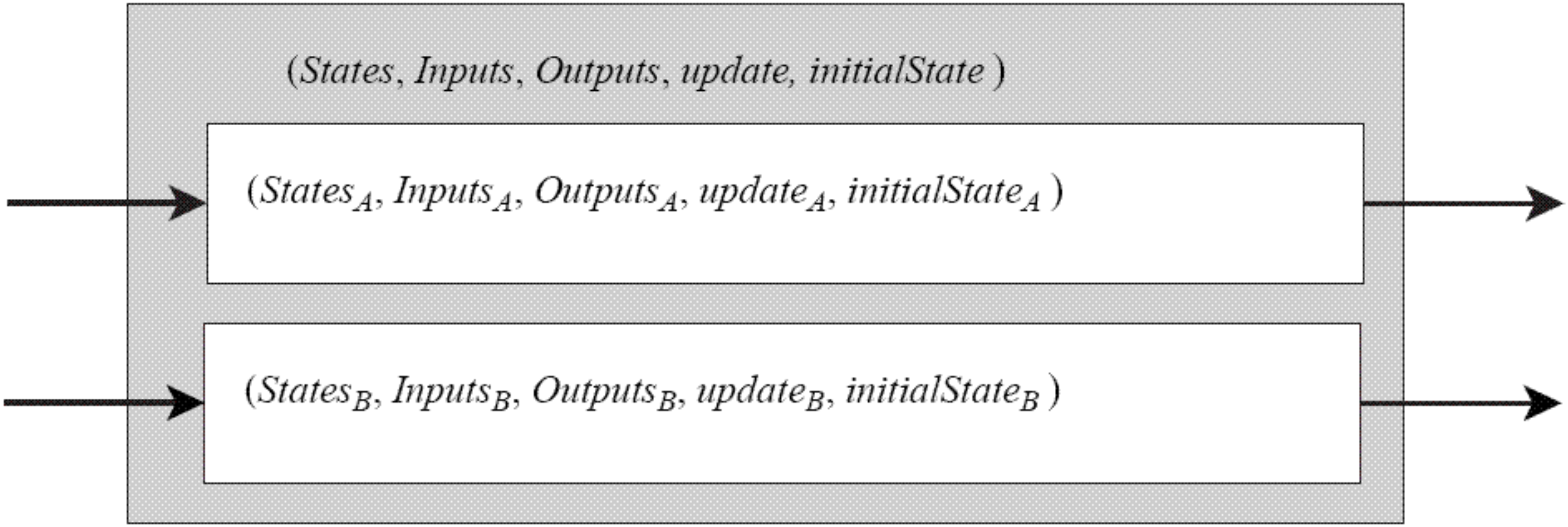
Side-by-Side Composition



A key question: When do these machines react?

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

Side-by-Side, Parallel Composition



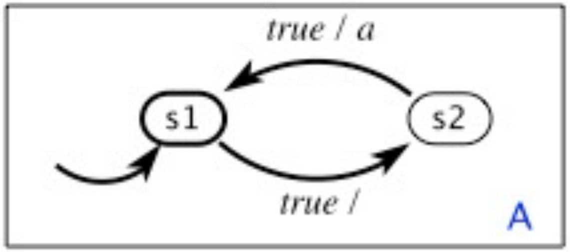
When do these machines react? Two of many possibilities:

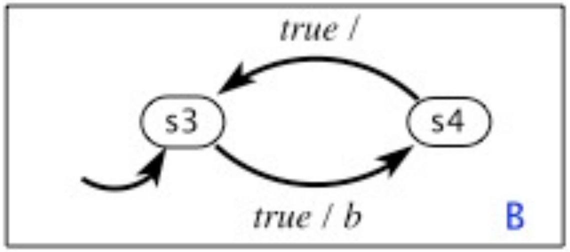
- Together, in lock step (synchronous, concurrent composition)
- Independently (asynchronous, concurrent composition)

Synchronous Composition

$$S_C \subseteq S_A \times S_B$$

outputs: a, b (pure)

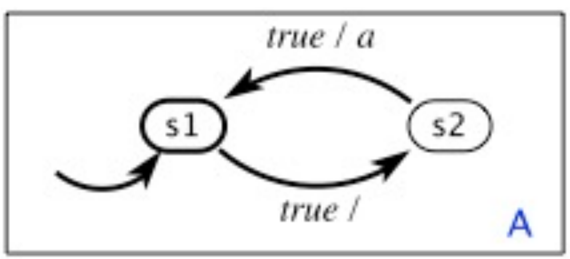


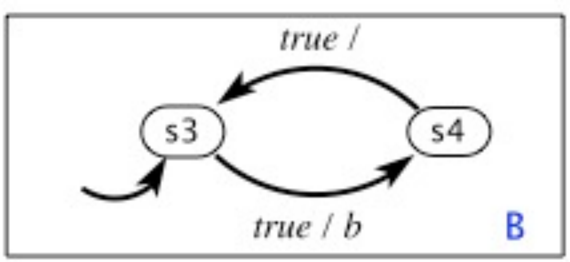


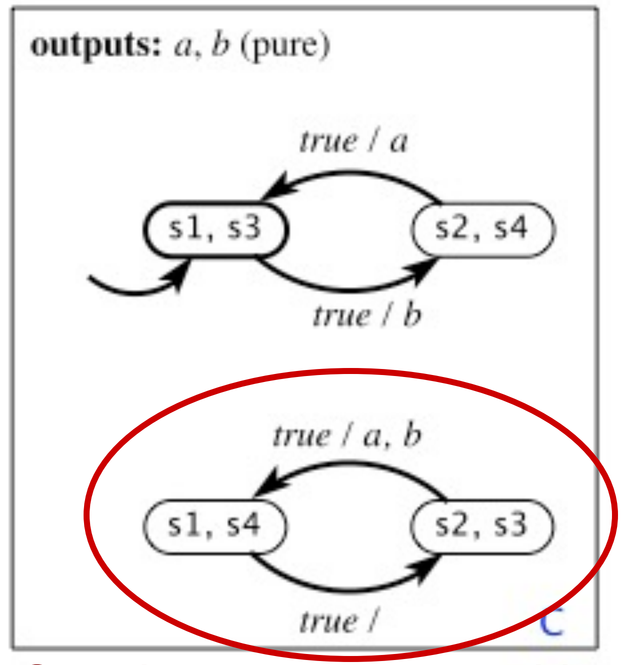
Synchronous Composition

$$S_C \subseteq S_A \times S_B$$

outputs: a, b (pure)







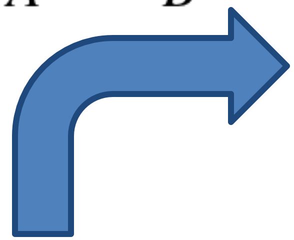
Synchronous composition

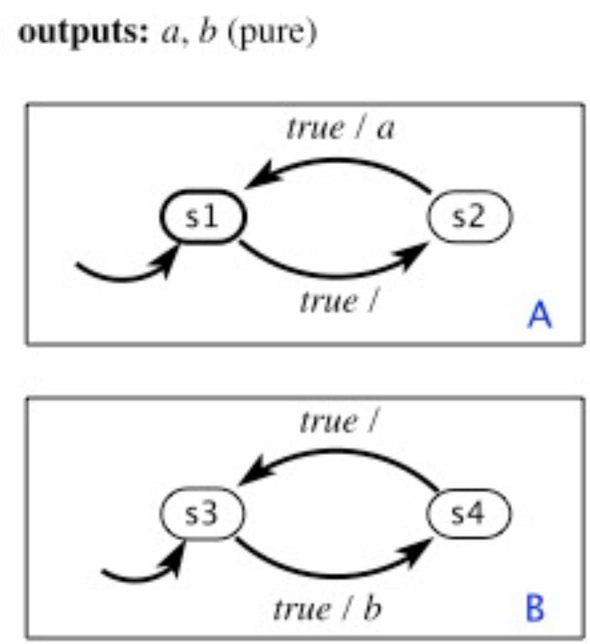
Note that these two states are not reachable.

Composition multiplies the state space

Asynchronous Composition

$$S_C \subseteq S_A \times S_B$$

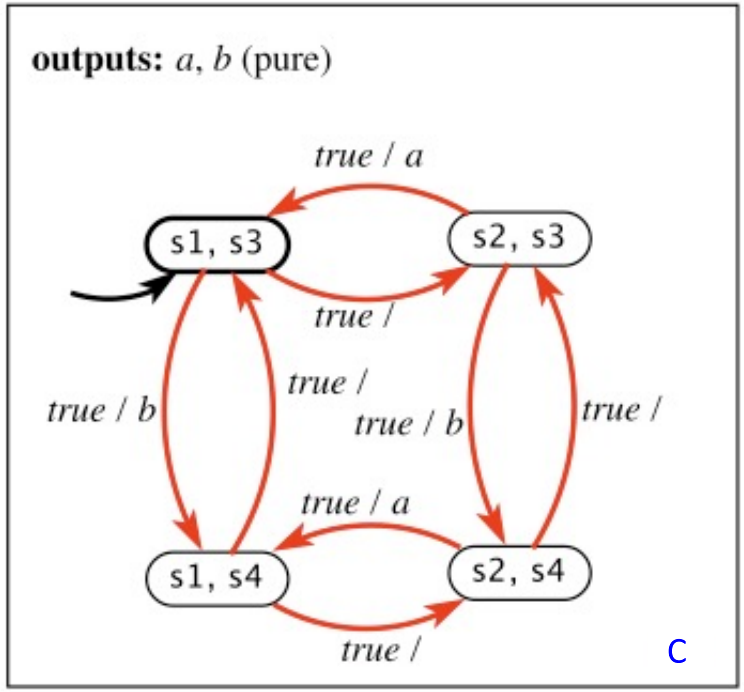




Asynchronous composition using <u>interleaving</u> semantics

Asynchronous Composition

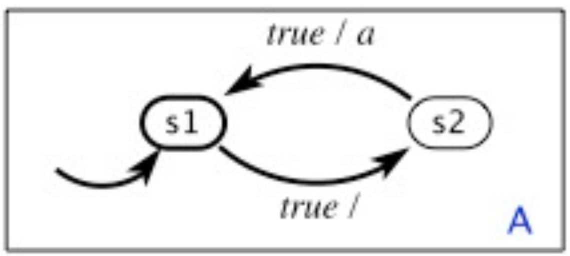
$$S_C \subseteq S_A \times S_B$$

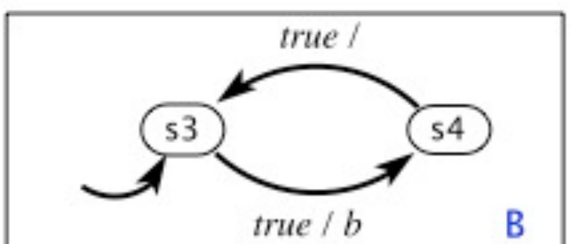


Asynchronous composition using <u>interleaving</u> semantics

Note that now all states are reachable.

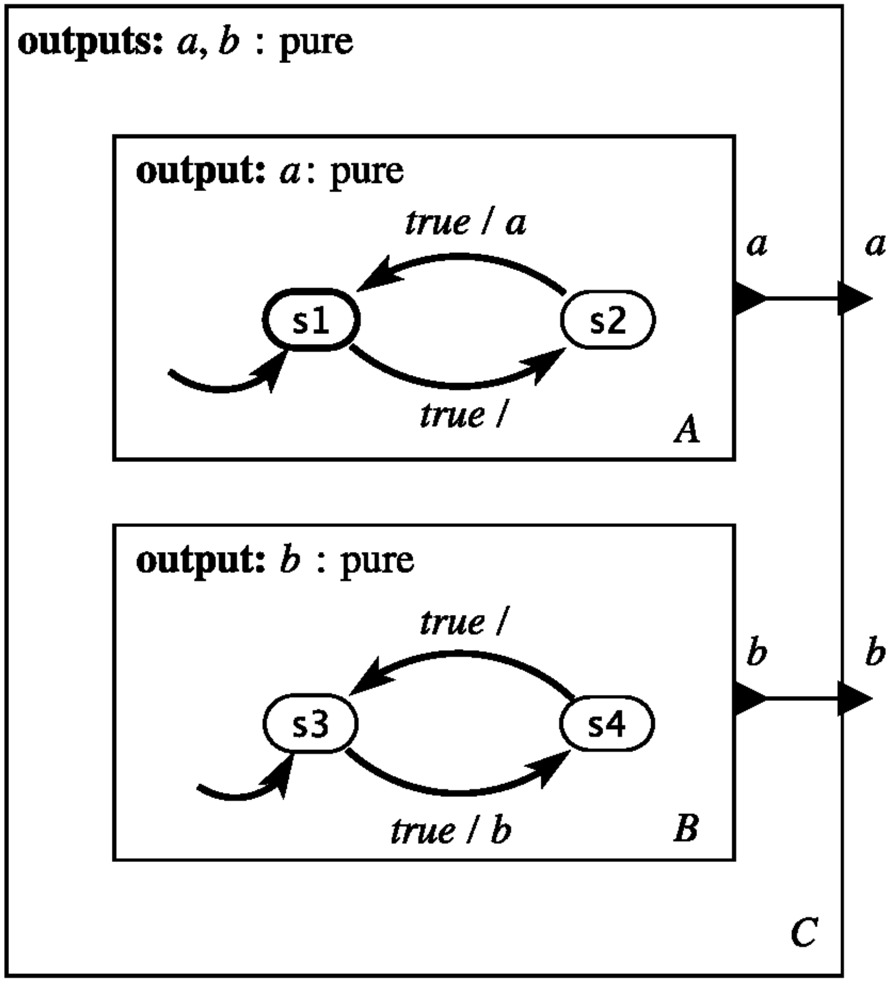
outputs: a, b (pure)





Syntax vs. Semantics

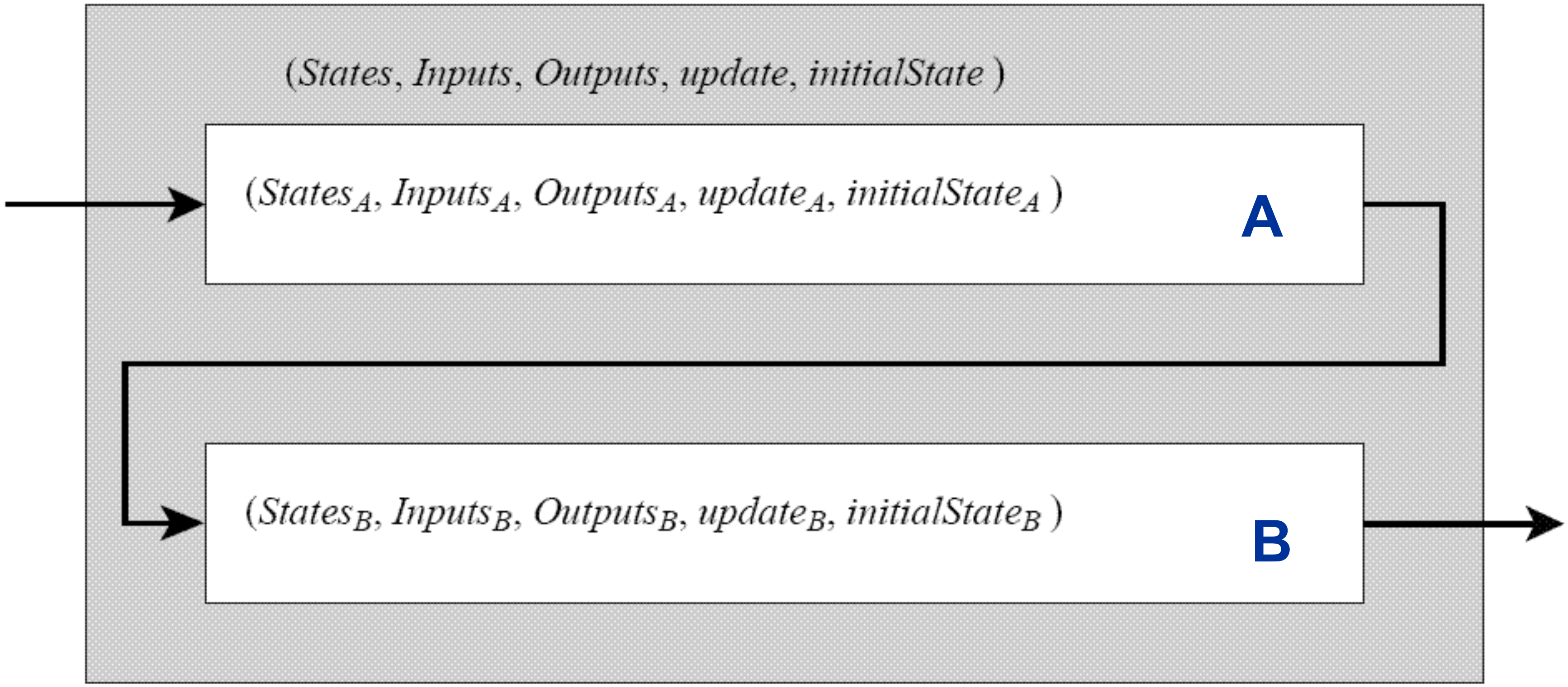
The answers to these questions defines the MoC being used.



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?

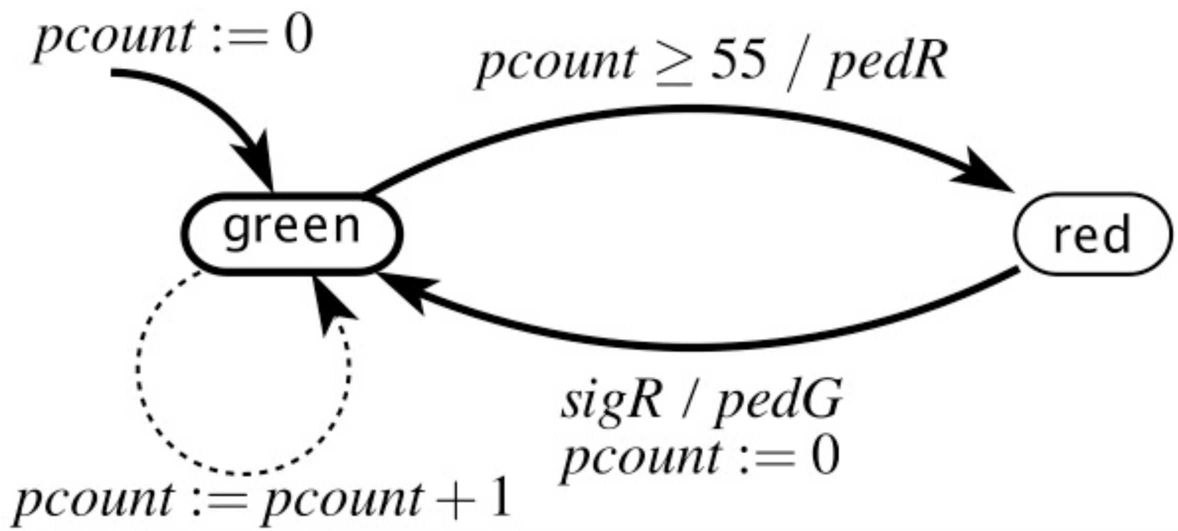
Cascade Composition



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

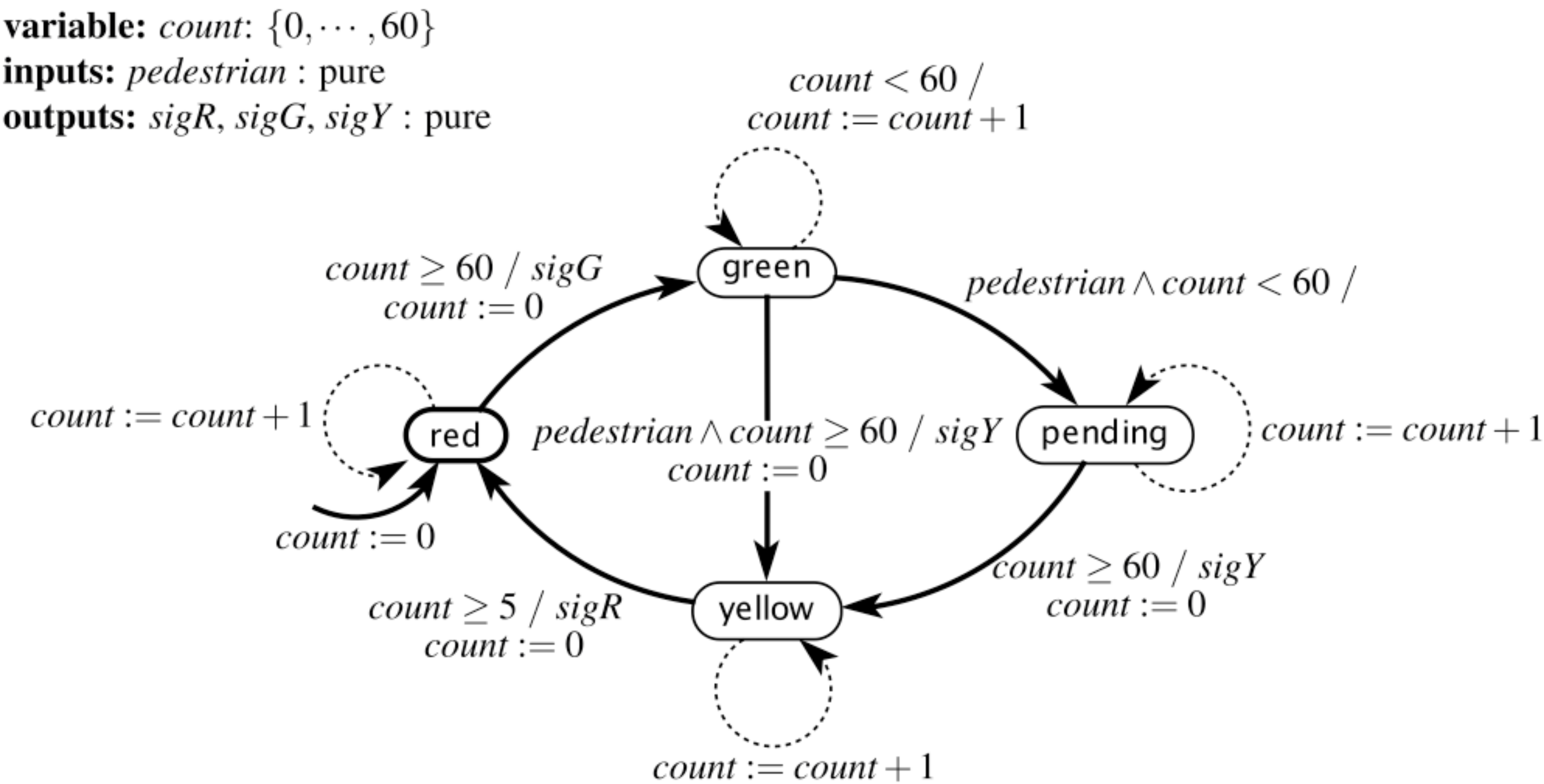
Example: Time-Triggered Pedestrian Light

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

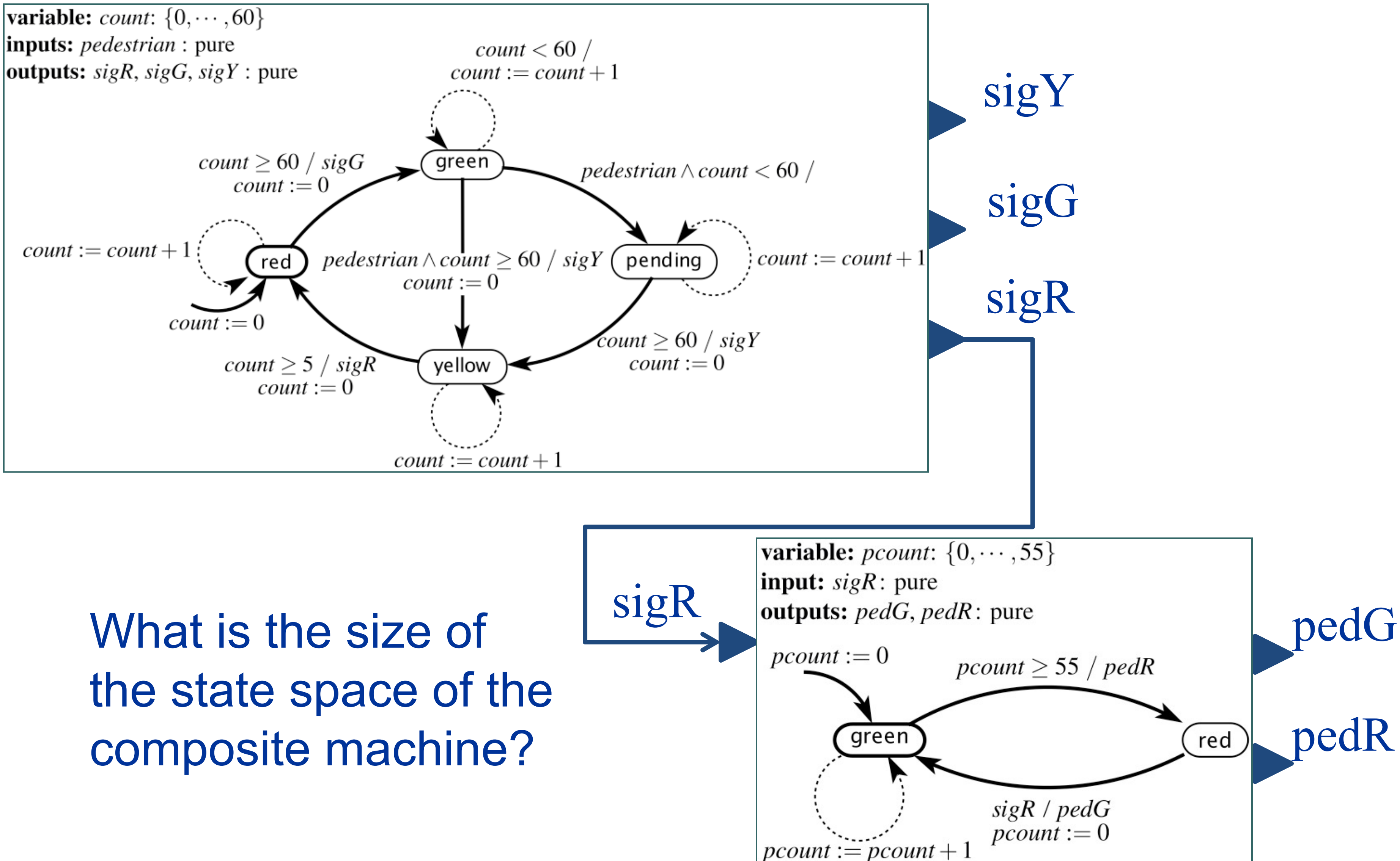


This light stays green for 55 seconds, then goes red. Upon receiving a sigR input, it repeats the cycle.

Example: Time-Triggered Car Light

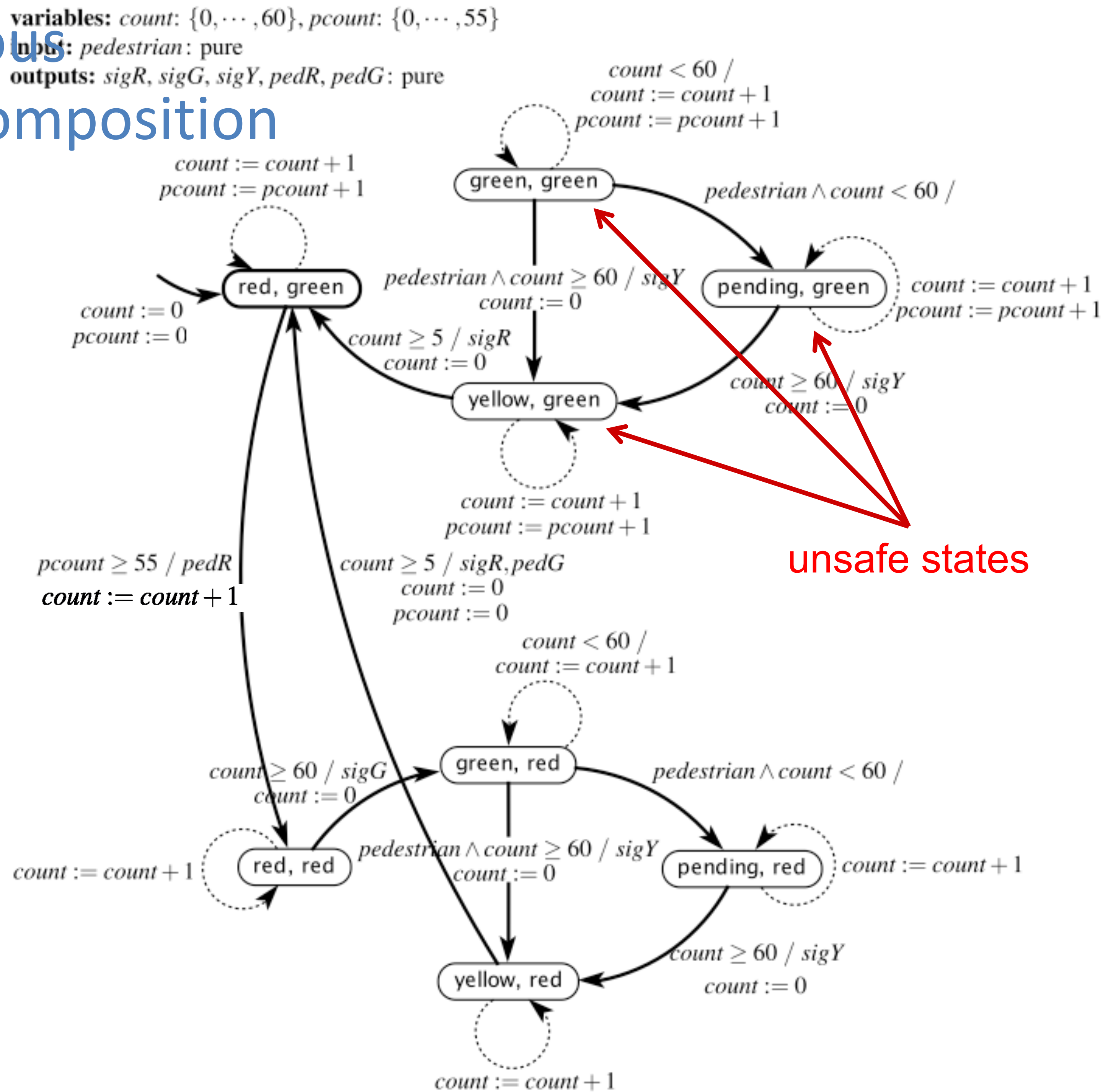


Pedestrian Light with Car Light



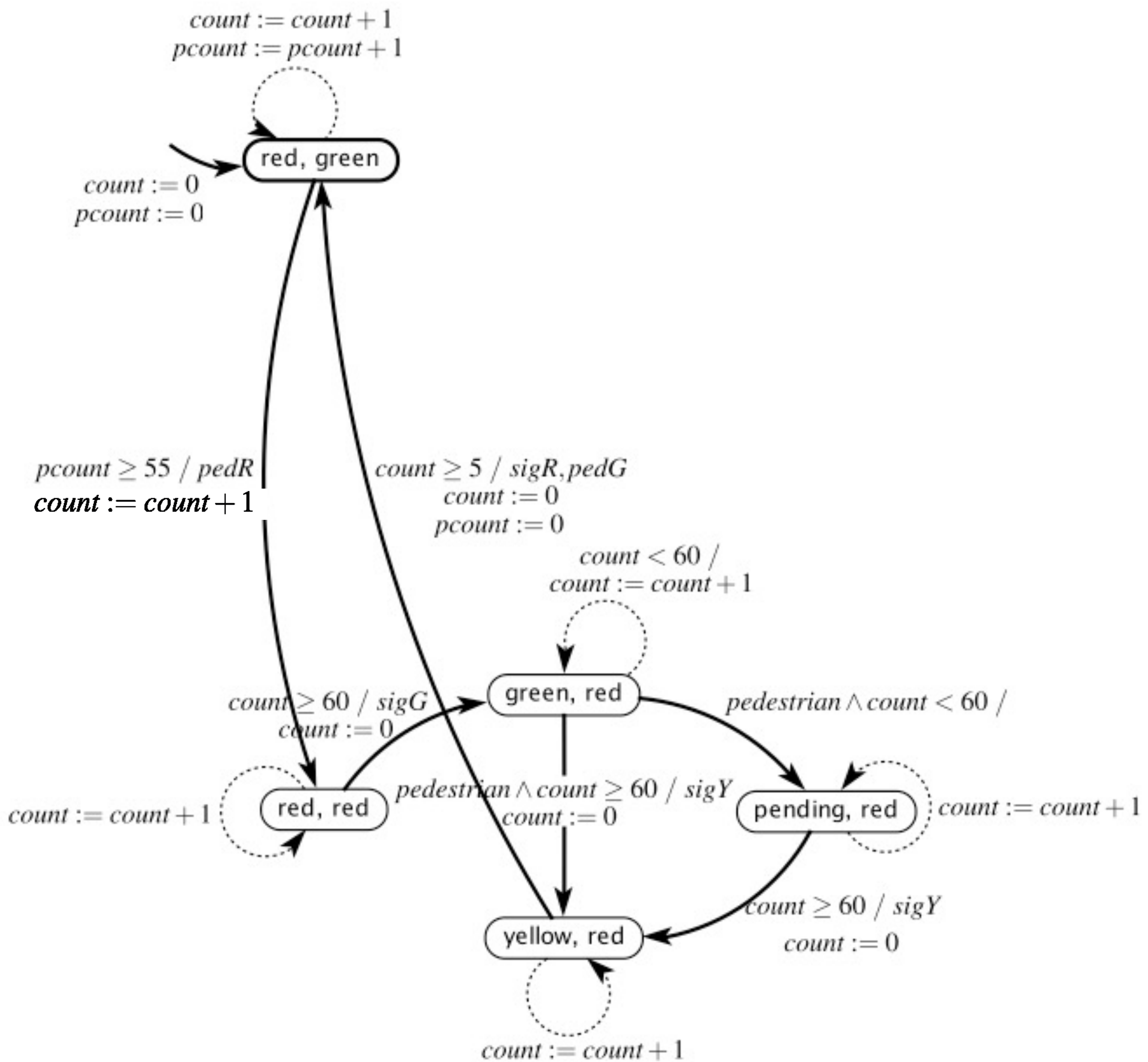
What is the size of the state space of the composite machine?

Synchronous cascade composition

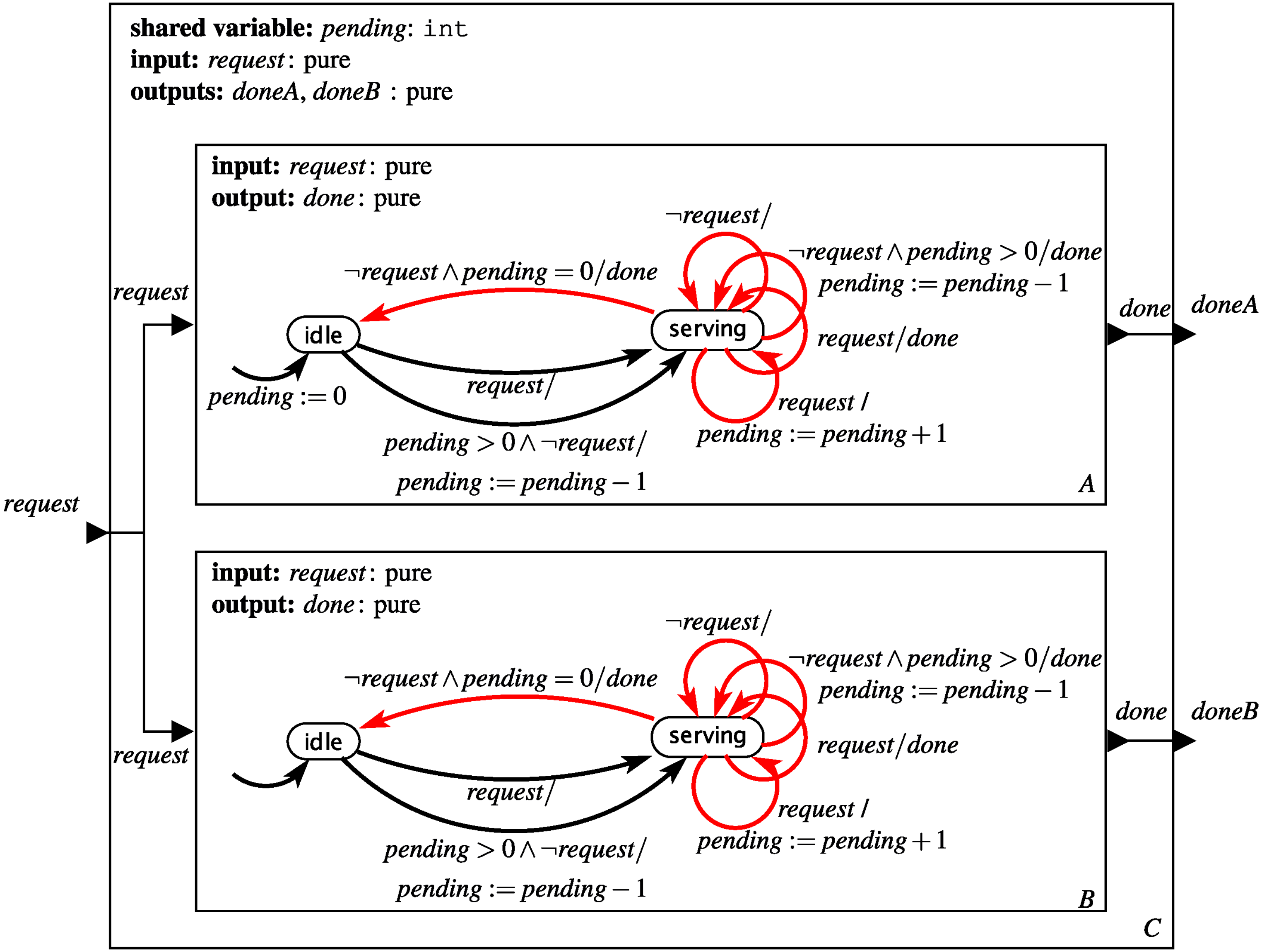


Unreachable states removed

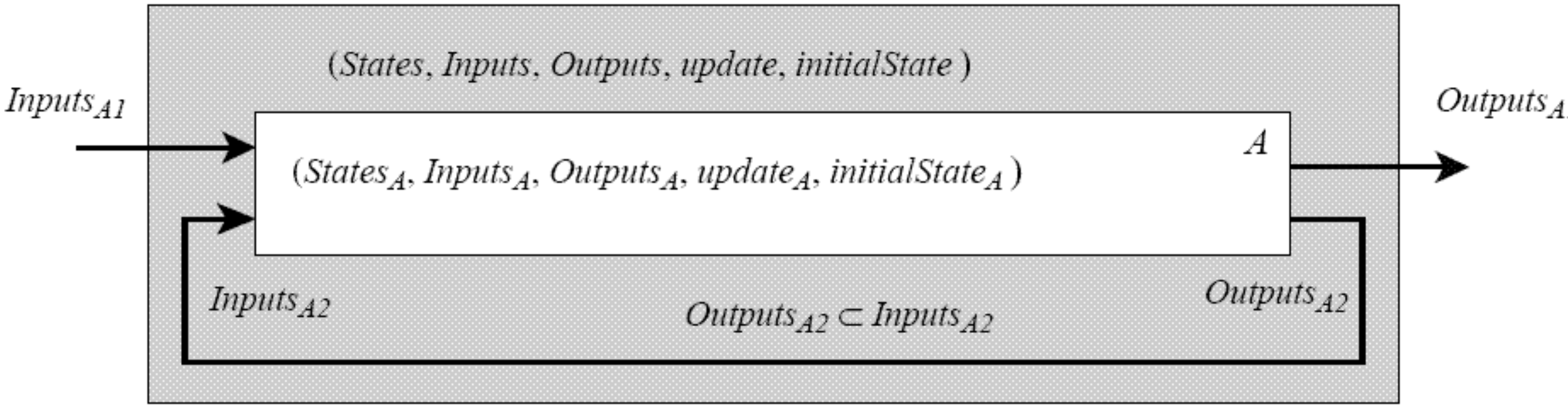
variables: count: $\{0,\cdots,60\}$, pcount: $\{0,\cdots,55\}$
input: pedestrian: pure
outputs: sigR, sigG, sigY, pedR, pedG: pure



Shared Variables: Two Servers



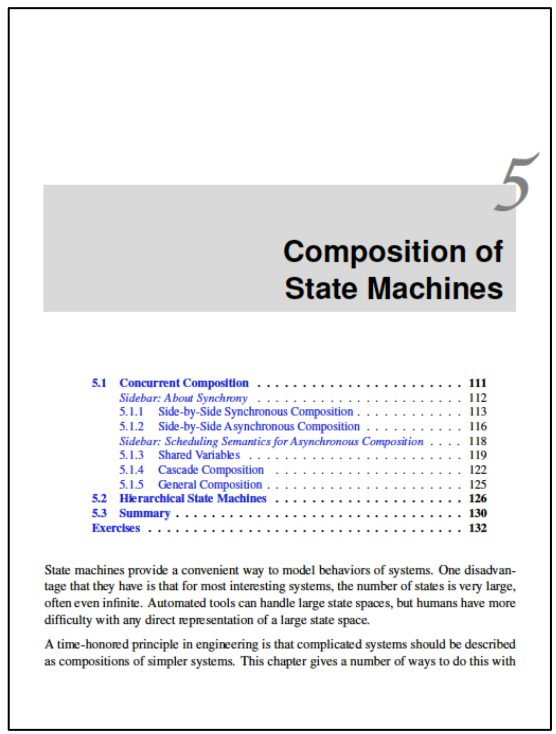
Feedback Composition



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

Things to do …

- If you haven't already done, read Chapter 5


5

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. Automated tools can handle large state spaces, but humans have more difficulty with any direct representation of a large state space.

A time-honored principle in engineering is that complicated systems should be described as compositions of simpler systems. This chapter gives a number of ways to do this with