Cyber-Physical Systems

Composite Models

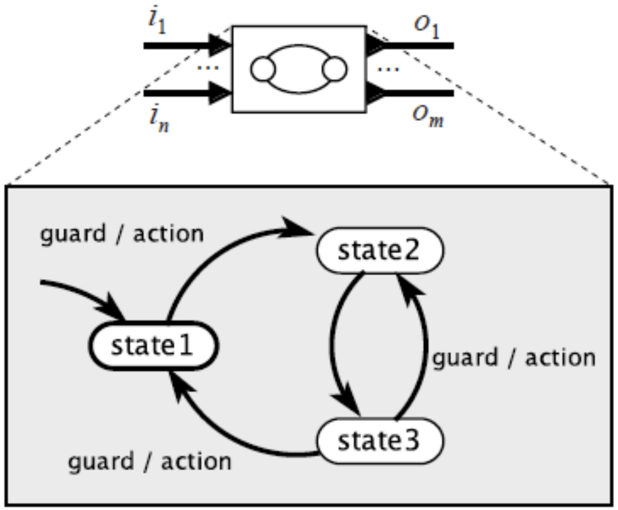
IECE 553/453– Fall 2022

Prof. Dola Saha

Composition of State Machines

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

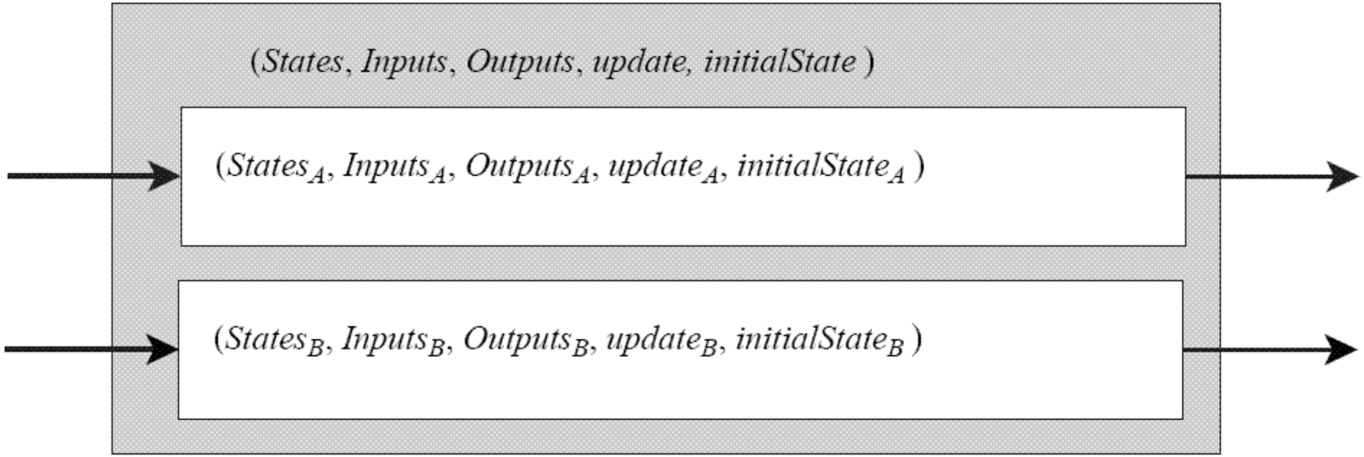
Expose inputs and outputs, enabling concurrent composition:



- Two kinds of composition:

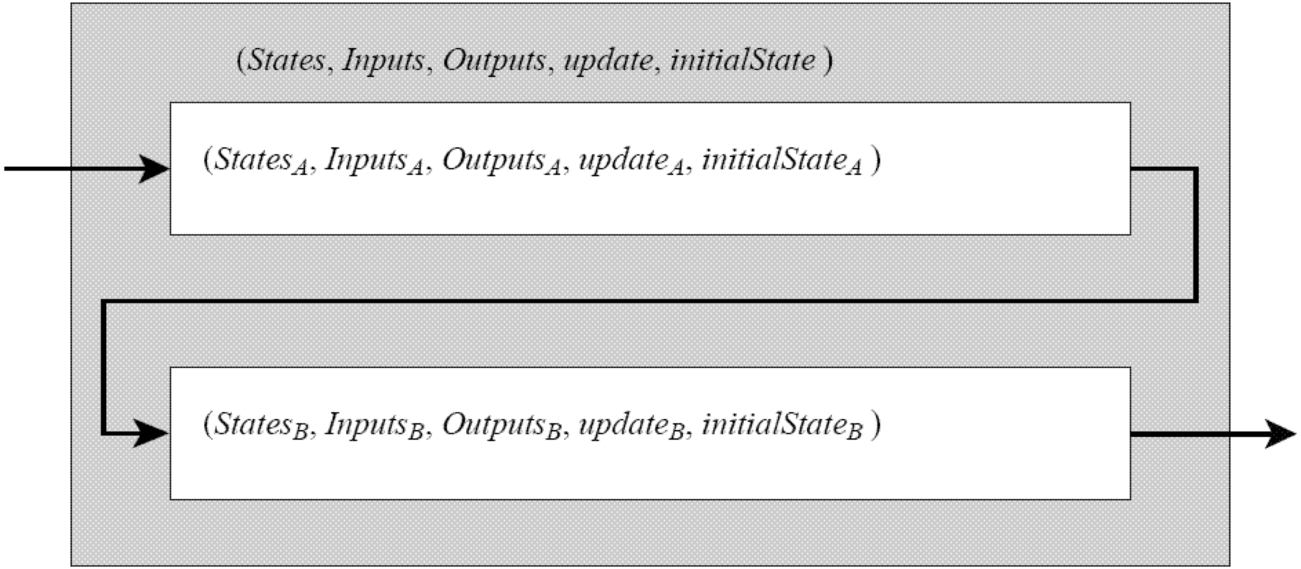
1. **Spatial**: how do the components communicate between each other?
2. **Temporal**: when do the components execute, relative to each other?

Side-by-Side Composition



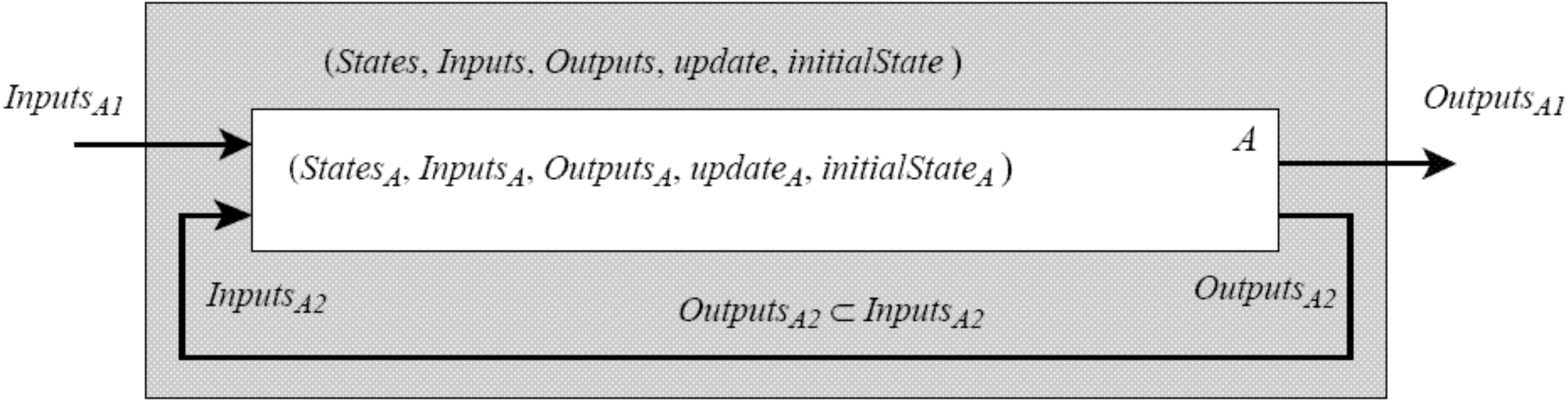
Synchronous composition: the machines react simultaneously and instantaneously.

Cascade Composition



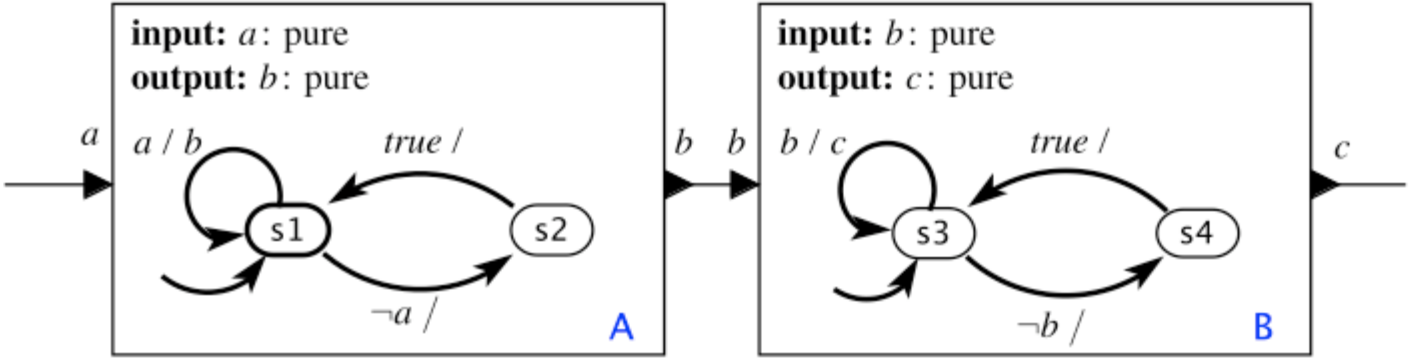
Synchronous composition: the machines react simultaneously and instantaneously, despite the apparent causal relationship!

Feedback Composition



Synchronous Composition

➢ Consider a cascade composition as follows:

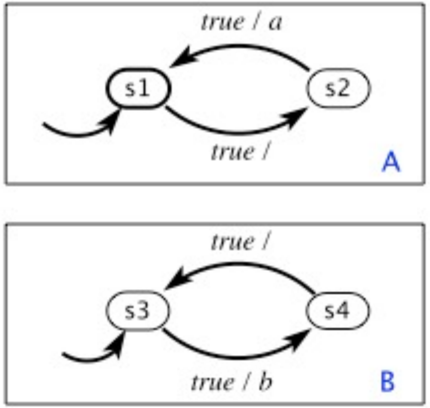


Reactions are *Simultaneous* and *Instantaneous*

Synchronous Composition

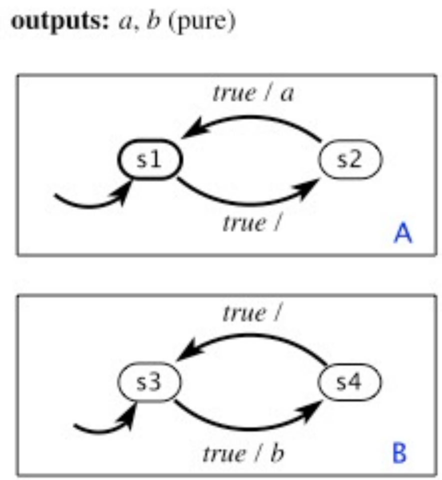
$$S_C \subseteq S_A \times S_B$$

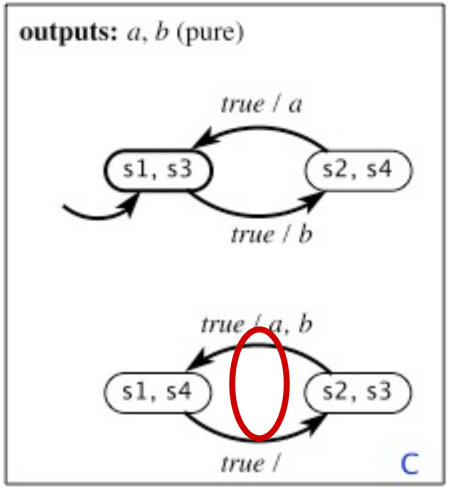
outputs: a, b (pure)



Synchronous Composition

$$S_C \subseteq S_A \times S_B$$





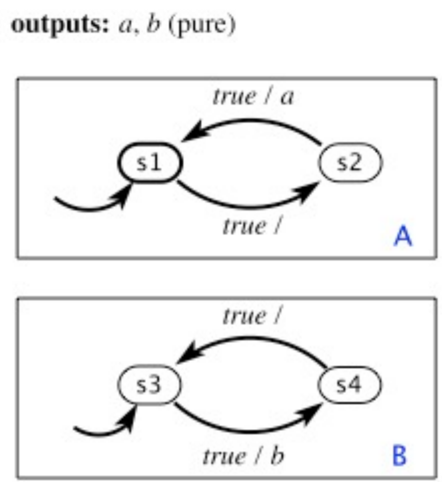
Synchronous composition

Note that these two states are not reachable.

Composition multiplies the state space

Asynchronous Composition

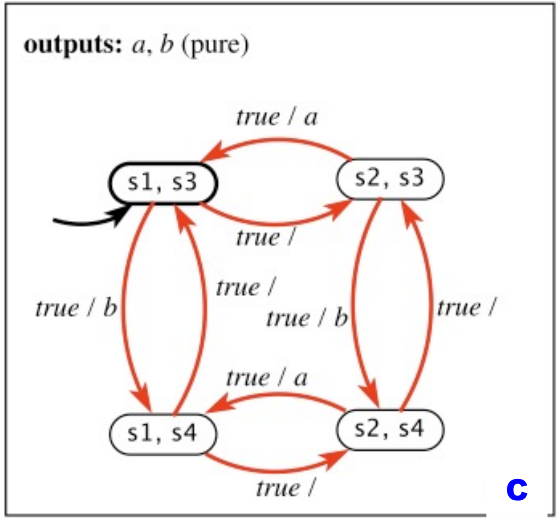
$$S_C \subseteq S_A \times S_B$$



Asynchronous composition using interleaving semantics

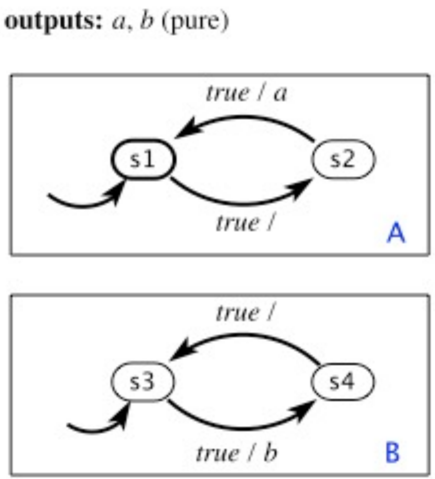
Asynchronous Composition

$$S_C \subseteq S_A \times S_B$$



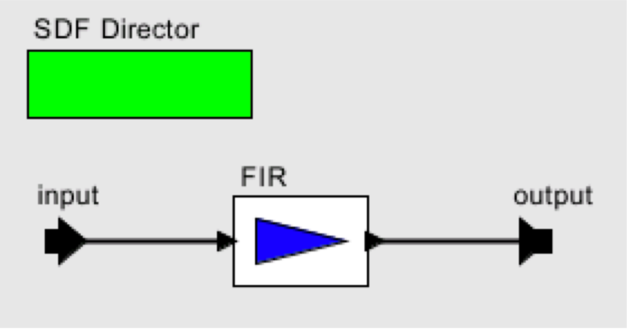
Asynchronous composition using <u>interleaving</u> semantics

Note that now all states are reachable.



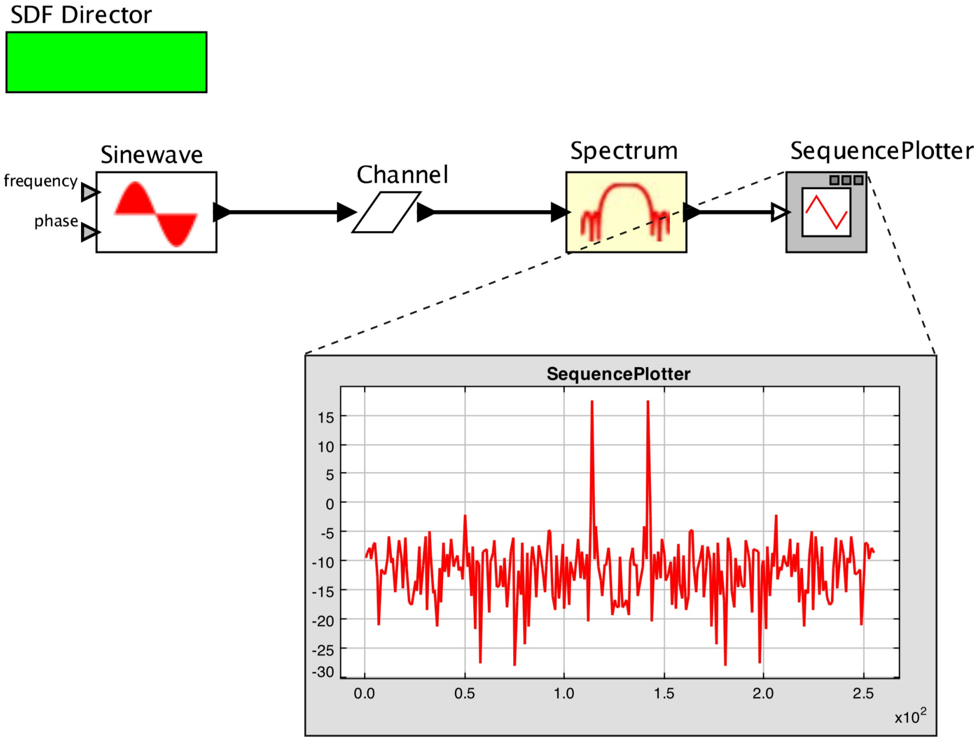
Synchronous Dataflow (SDF)

- Specialized model for dataflow
- All actors consume input tokens, perform their computation and produce outputs in one atomic operation
- Flow of control is known (predictable at compile time)
- Statically scheduled domain
- Useful for synchronous signal processing systems
- Homogeneous SDF: one token is usually produced for every iteration

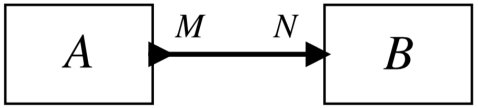


Multirate SDF Model

- The firing rates of the actors are not identical
- The Spectrum actor requires 256 tokens to fire, so one iteration of this model requires 256 firings of Sinewave, Channel, and SequencePlotter, and one firing of Spectrum.



Balance Equations

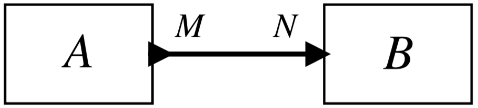


- When A fires, it produces M tokens on its output port
- When B fires, it consumes N tokens on its input port
- M and N are non-negative integers
- Suppose that A fires q_A times and B fires q_B times
- All tokens that A produces are consumed by B if and only if the following **balance equation** is satisfied

$$q_A M = q_B N$$

- The system remains in balance if and only if the balance equation is satisfied

Example



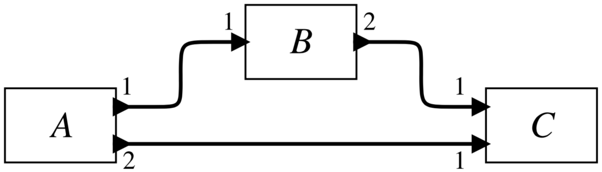
- Suppose M=2, N=3
- Possible Solution:
  - $q_A=3$, $q_B=2$
  - Example Schedule : {A, A, A, B, B} OR {A, B, A, A, B}
- Another Possible Solution:
  - $q_A=6$, $q_B=4$
  - Example Schedule: {A,A,A,A,A,A,B,B,B,B}

Strategy for firing

- Streaming applications: arbitrarily large number of tokens
- Naive strategy: fire actor A an arbitrarily large number q_A times, and then fire actor B q_B times
  - Why naive?
- Better strategy:
  - smallest positive q_A and q_B that satisfy the balance equation
- Unbounded execution with bounded buffers

Solving the Balance Equation

- Every connection between actors results in a balance equation
- The model defines a system of equations, and the goal is to find the least positive integer solution

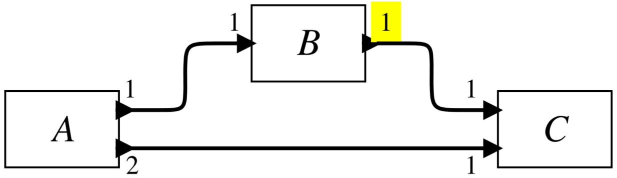


$$
\begin{aligned}
q_A &= q_B \\
2q_B &= q_C \\
2q_A &= q_C
\end{aligned}
$$

- The *least* positive integer solution to these equations is
  - $q_A = q_B = 1$, and $q_C = 2$
- The schedule {A, B, C, C} can be repeated forever to get an unbounded execution with bounded buffers

Fig 6.9 from Book

Inconsistent SDF



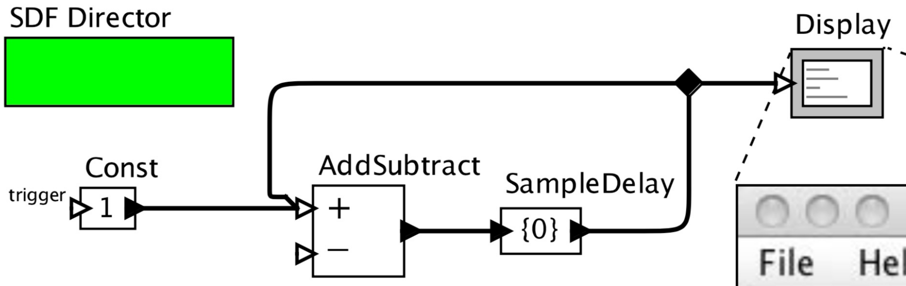
$$q_A = q_B = q_C = 0$$

- An SDF model that has a non-zero solution to the balance equations is said to be consistent.
- If the only solution is zero, then it is inconsistent.
- An inconsistent model has no unbounded execution with bounded buffers.

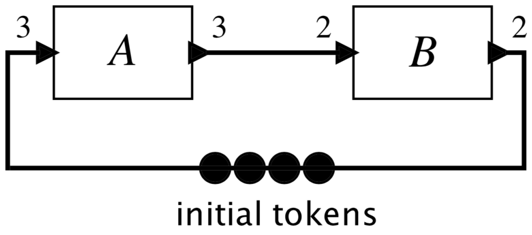
Fig 6.10 from Book

Feedback Loop

- A feedback loop in SDF must include at least one instance of the SampleDelay actor
- Without this actor, the loop would deadlock
  - actors in the feedback loop would be unable to fire because they depend on each other for tokens.
- The initial tokens enable downstream actors to fire and break the circular dependencies that would otherwise result from a feedback loop



Example Feedback Loop

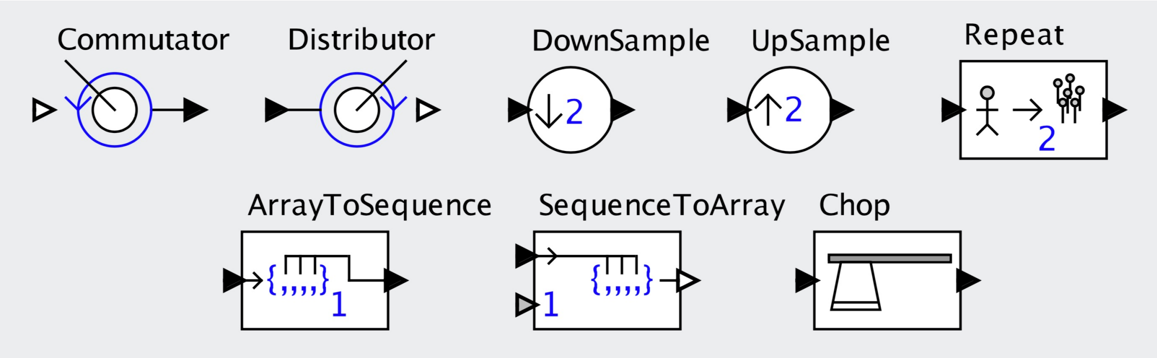


$$3q_A = 2q_B$$
$$2q_B = 3q_A \quad A, B, A, B, B$$

- The least positive integer solution is
  - qA = 2, qB = 3, so the model is consistent.
- With 4 initial tokens: consistent
- With 3 initial tokens: deadlock
  - If there were only three tokens, then A could fire, followed by B, but neither would have enough input tokens to fire again.

Multirate Dataflow Actors

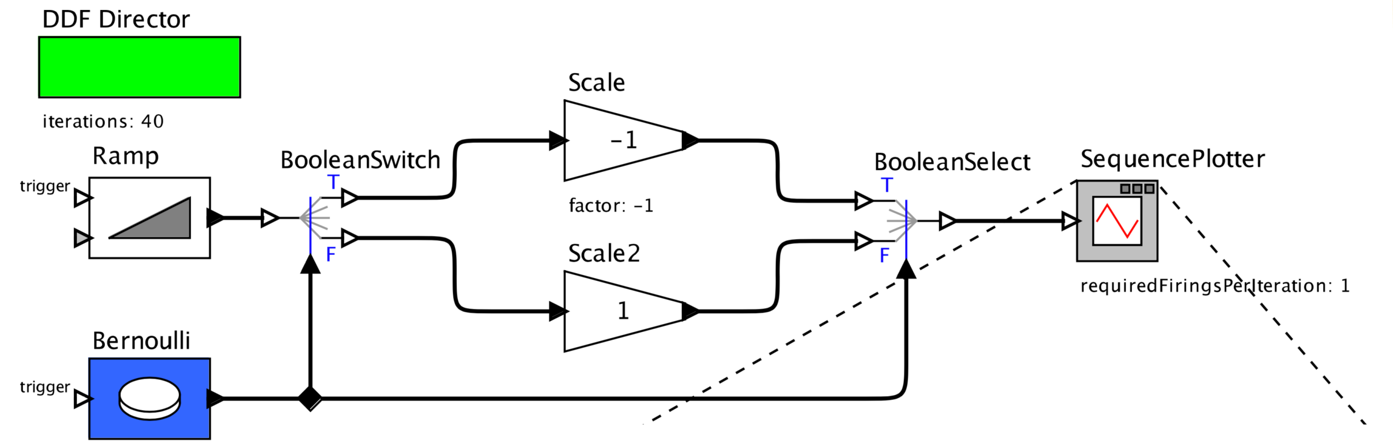
- actors that produce and/or consume multiple tokens per firing on a port



Dynamic Dataflow (DDF)

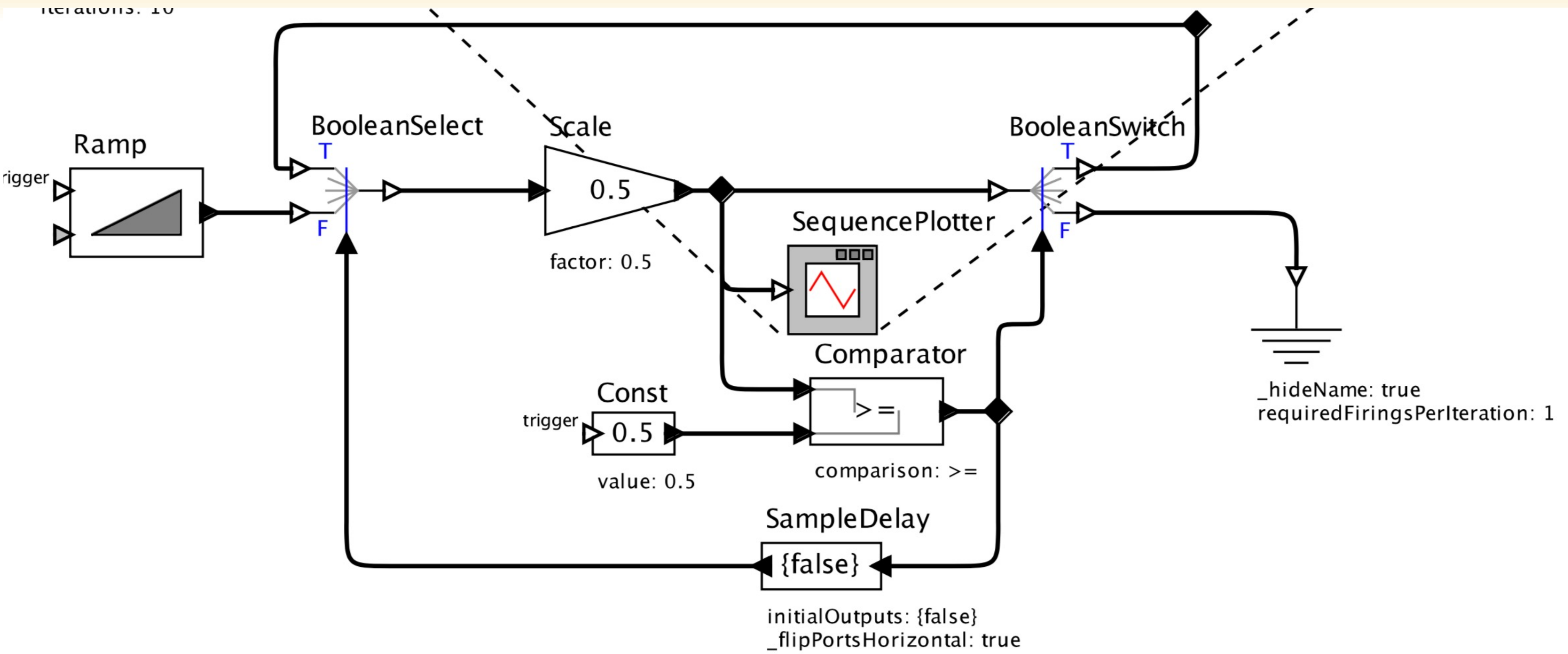
- SDF cannot express conditional firing: an actor fires only if a token has a particular value
- DDF: Firing Rule is required to be satisfied for firing
- Number of tokens produced can vary
- Example DDF Actor: Select
- Similar to Go To in Imperative Programming

Example DDF (Conditional Firing)

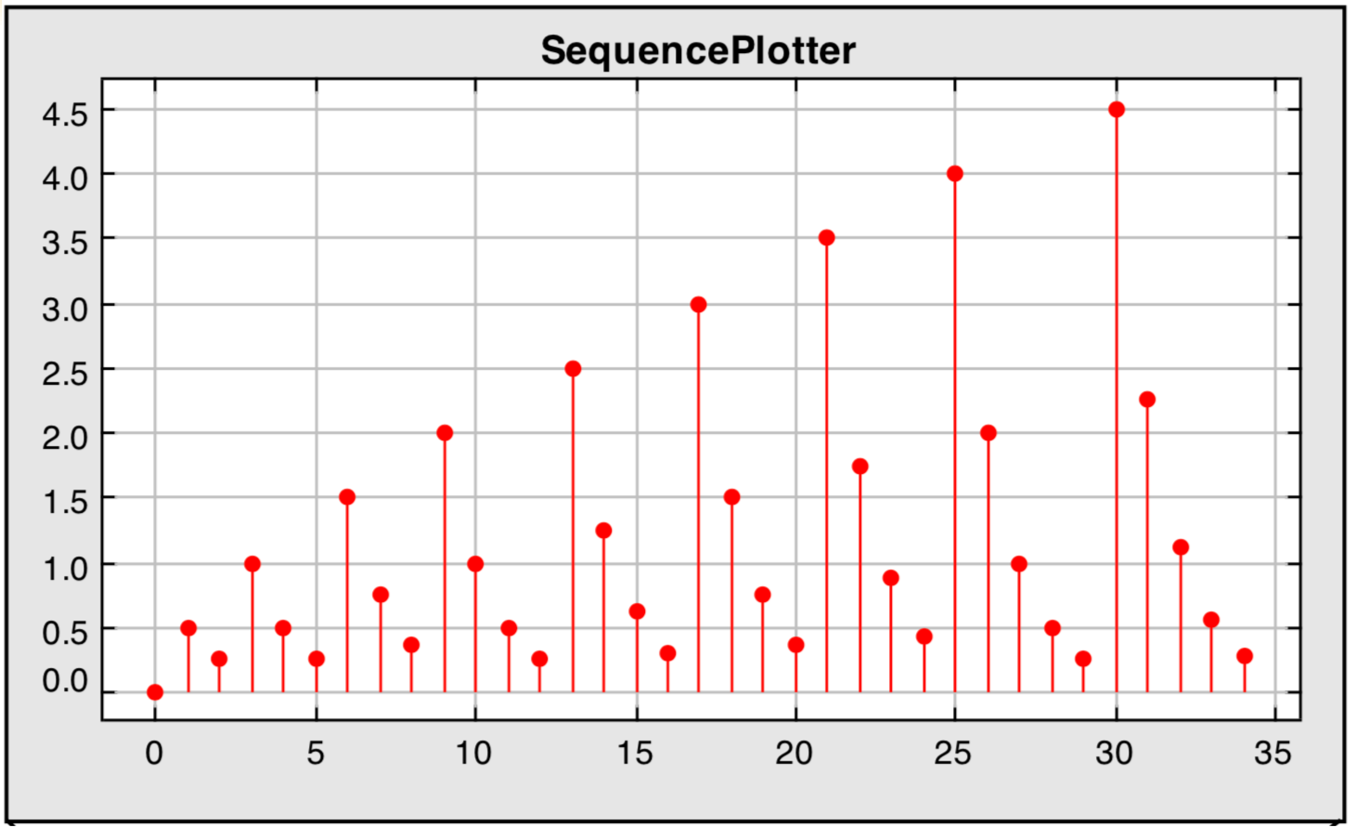


When Bernoulli produces true, the output of the Ramp actor is multiplied by −1

Data Dependent Iteration

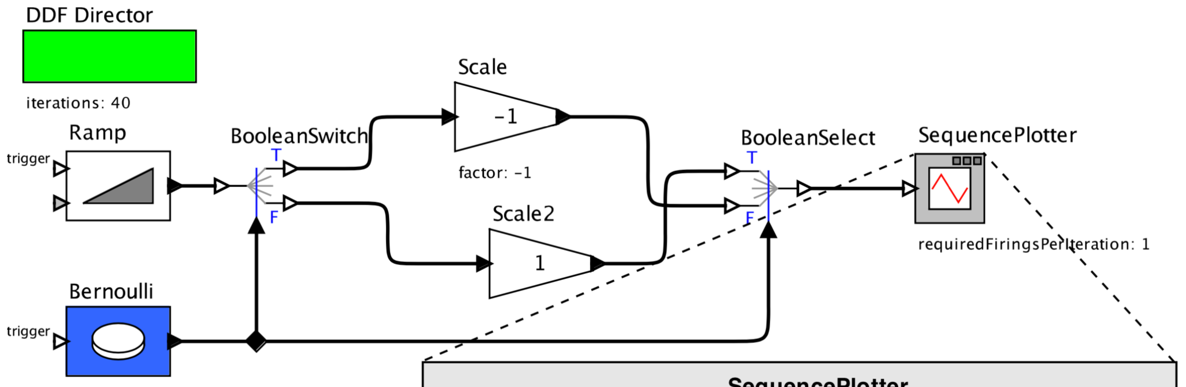


Conditional Firing Output



Unbounded Buffer Schedule

- The Bernoulli actor is capable of producing an arbitrarily long sequence of true-valued tokens, during which an arbitrarily long sequence of tokens may build up on input buffer for the *false* port of the BooleanSelect, thus potentially overflowing the buffer.

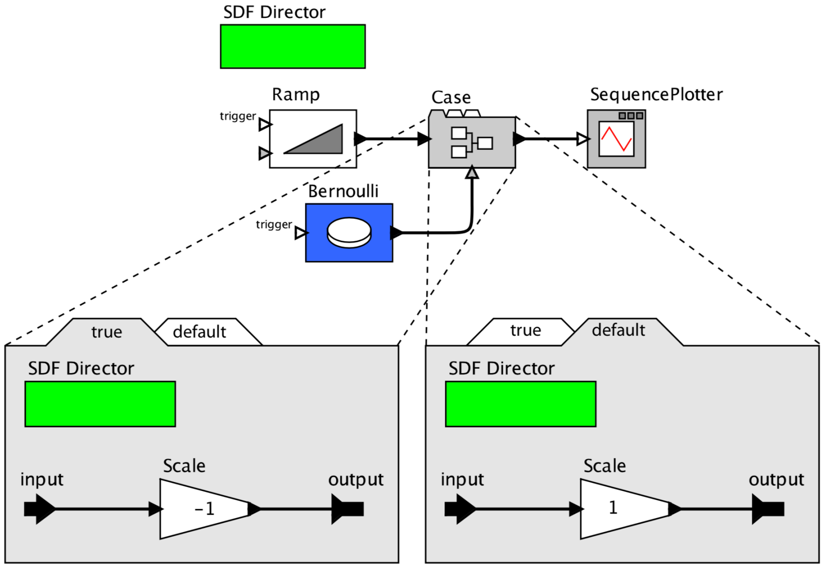


DDF

- It may not be possible to determine a schedule with bounded buffers
- Not always possible to ensure that the model will not deadlock
- Buck (1993) showed that bounded buffers and deadlock are undecidable for DDF models.
- DDF models are not as readily analyzed.
- Structured dataflow & higher order actors are used

Structured Dataflow

- Higher order actor: combine multiple actors as components
- Example Case: 2 sub-models
  - true that contains a Scale actor with a parameter of −1, and
  - default that contains a Scale actor with a parameter of 1.
  - When the control input to the Case actor is true, the true refinement executes one iteration. For any other control input, the default refinement executes.



Actor Model Implementation

- Multiple clocks
- Multiple domains
- Buffer: Queue
- Message: Interprocess communication