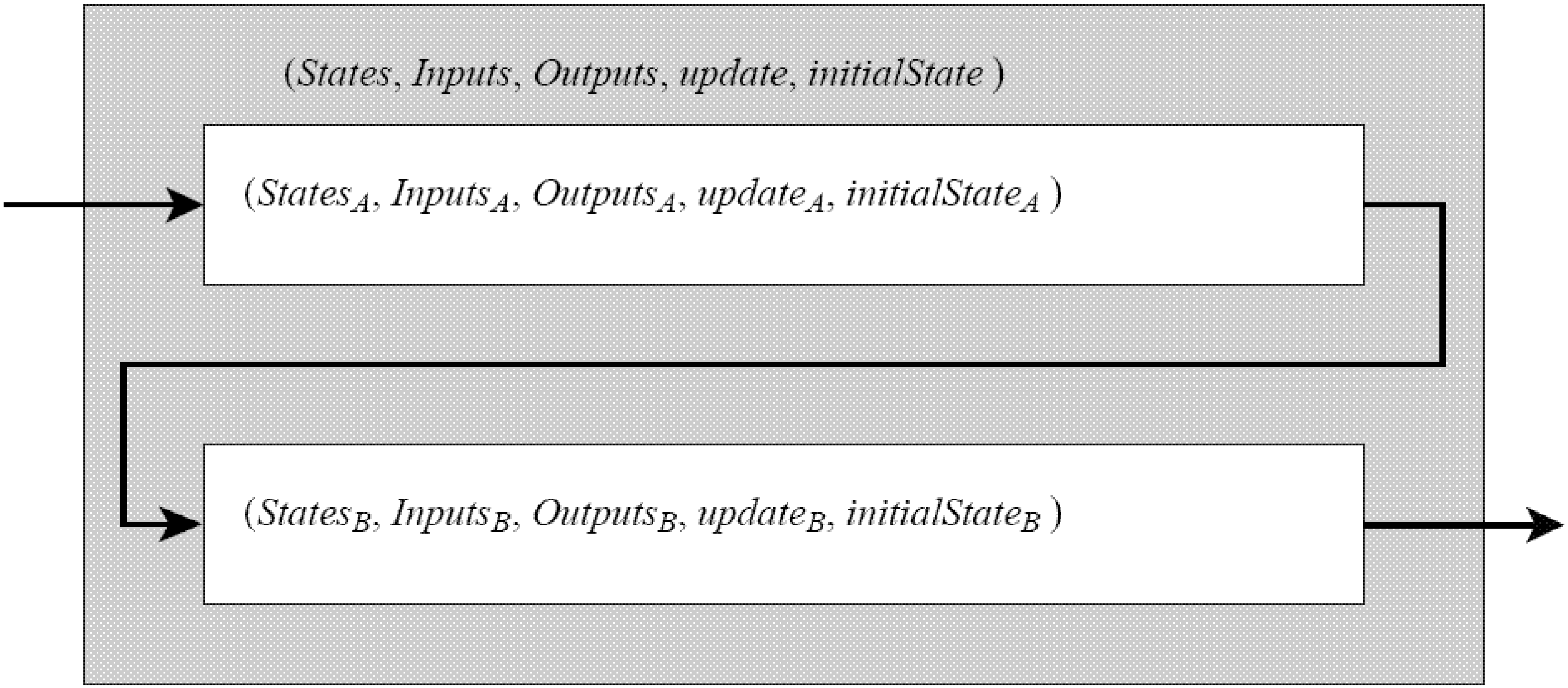
Embedded Systems 4

Synchronous Composition

Lee/Seshia
Section 6.2

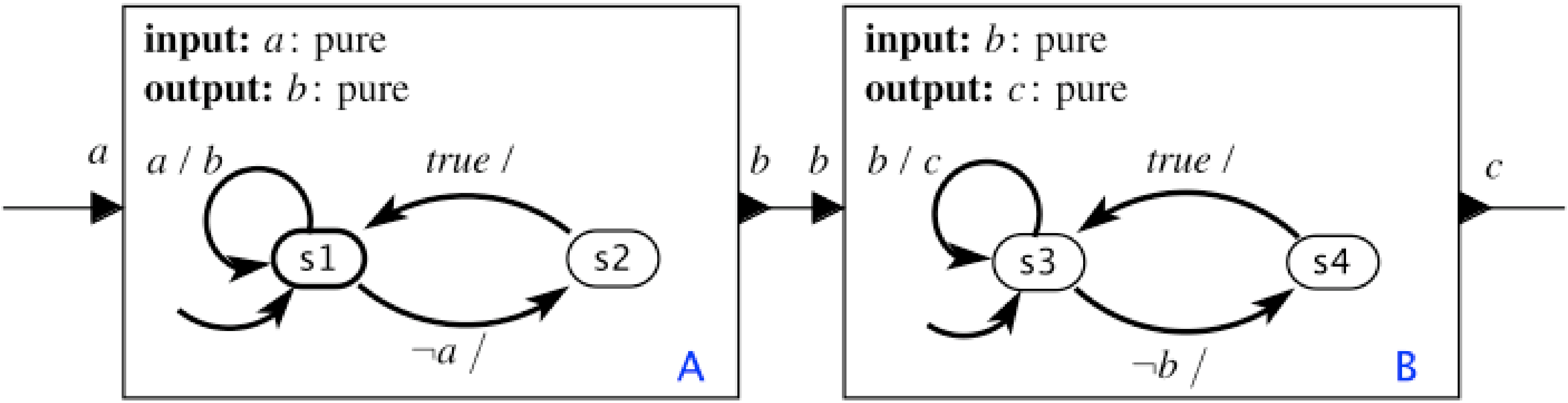
- Important semantic model for concurrent composition
- Here: composition of actors
- Foundation of Statecharts, Simulink, synchronous programming languages
  - Esterel
  - Lustre
  - Scade
- Idealistic view of concurrency, not adequate for distributed systems (Implicit assumption: presence of global clock and instant communication; requires broadcast mechanism)

Synchronous composition



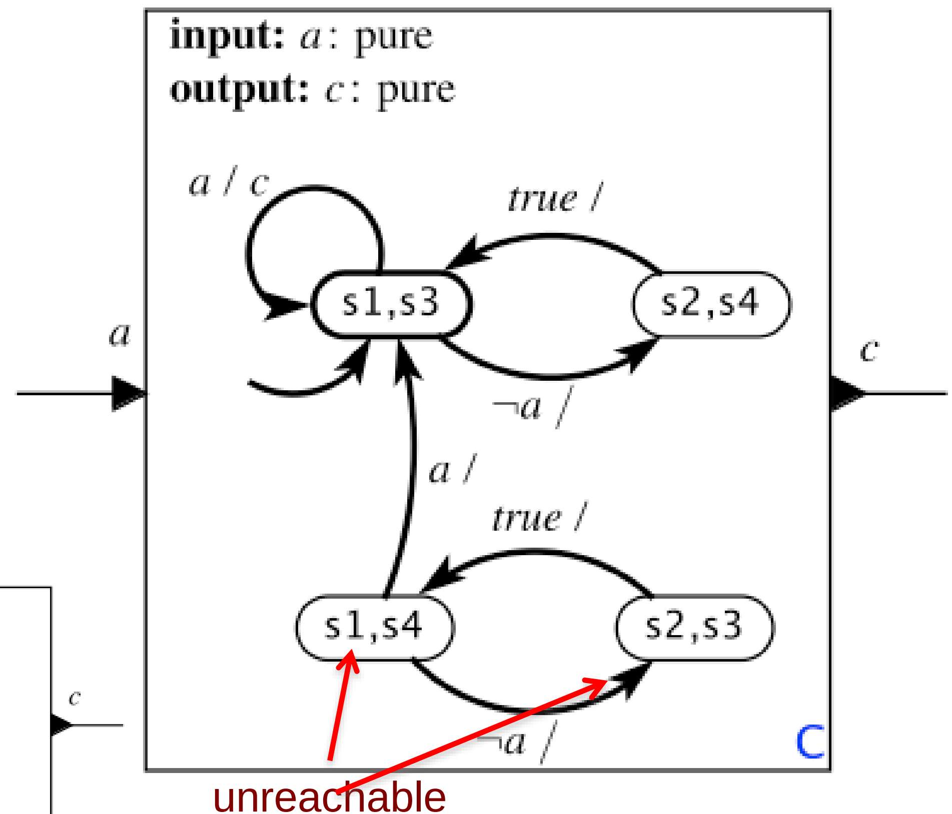
Synchronous composition: the machines react simultaneously and instantaneously, despite the apparent causal relationship!

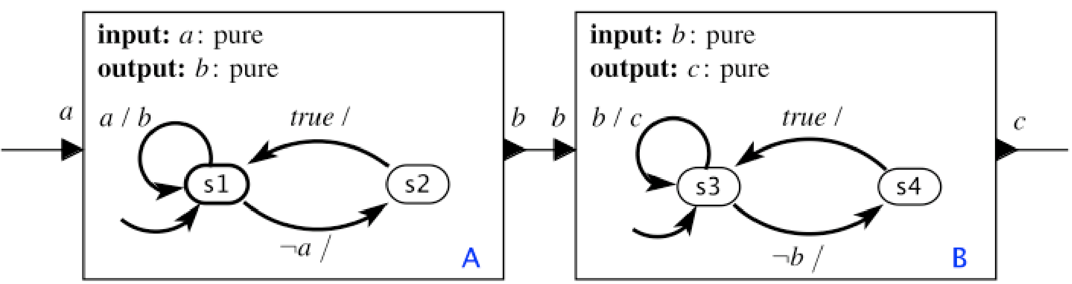
Synchronous composition:
Reactions are *simultaneous* and *instantaneous*



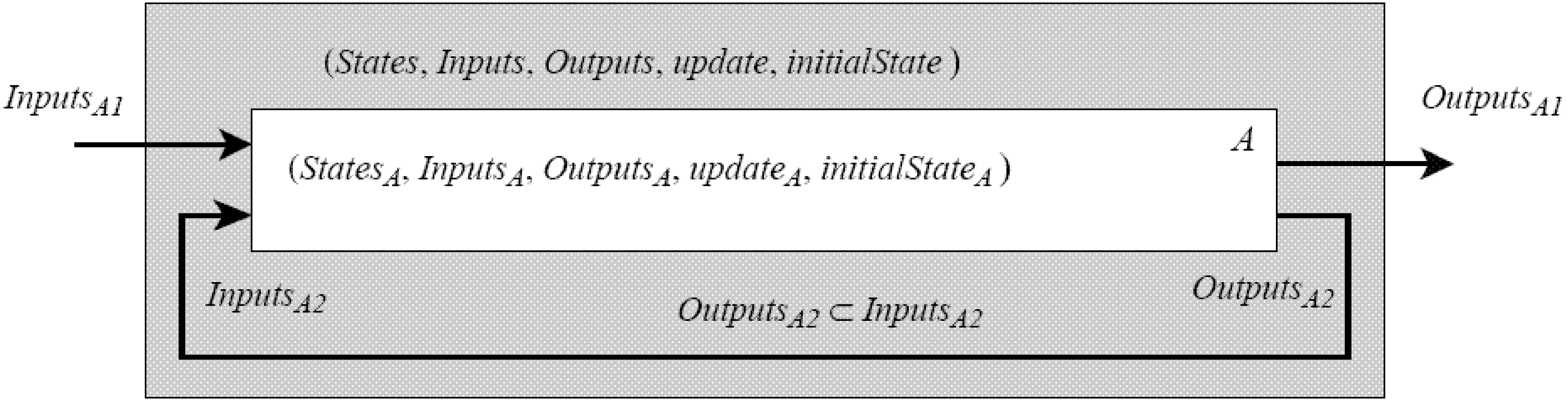
Synchronous composition:
Reactions are *simultaneous* and *instantaneous*

$$S_C = S_A \times S_B$$

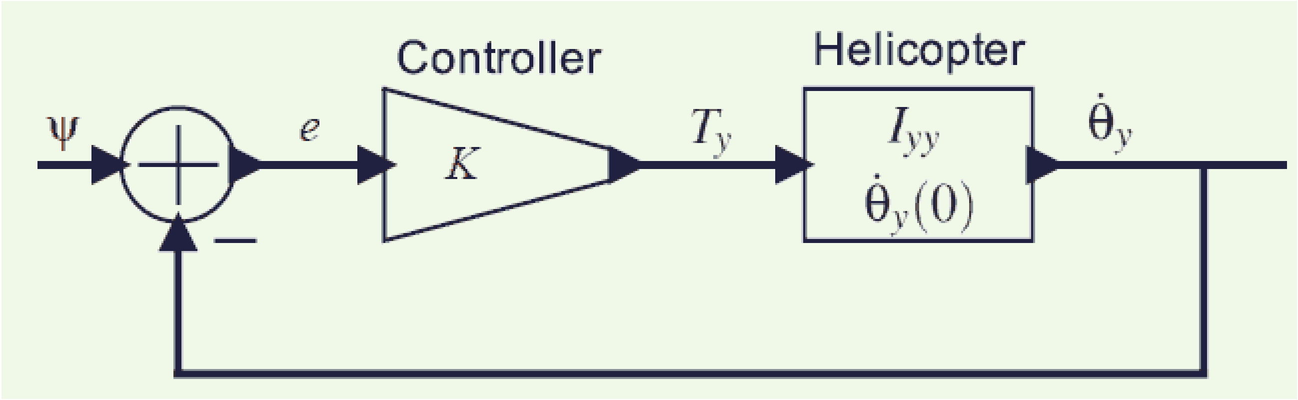




Feedback composition



Continuous feedback composition

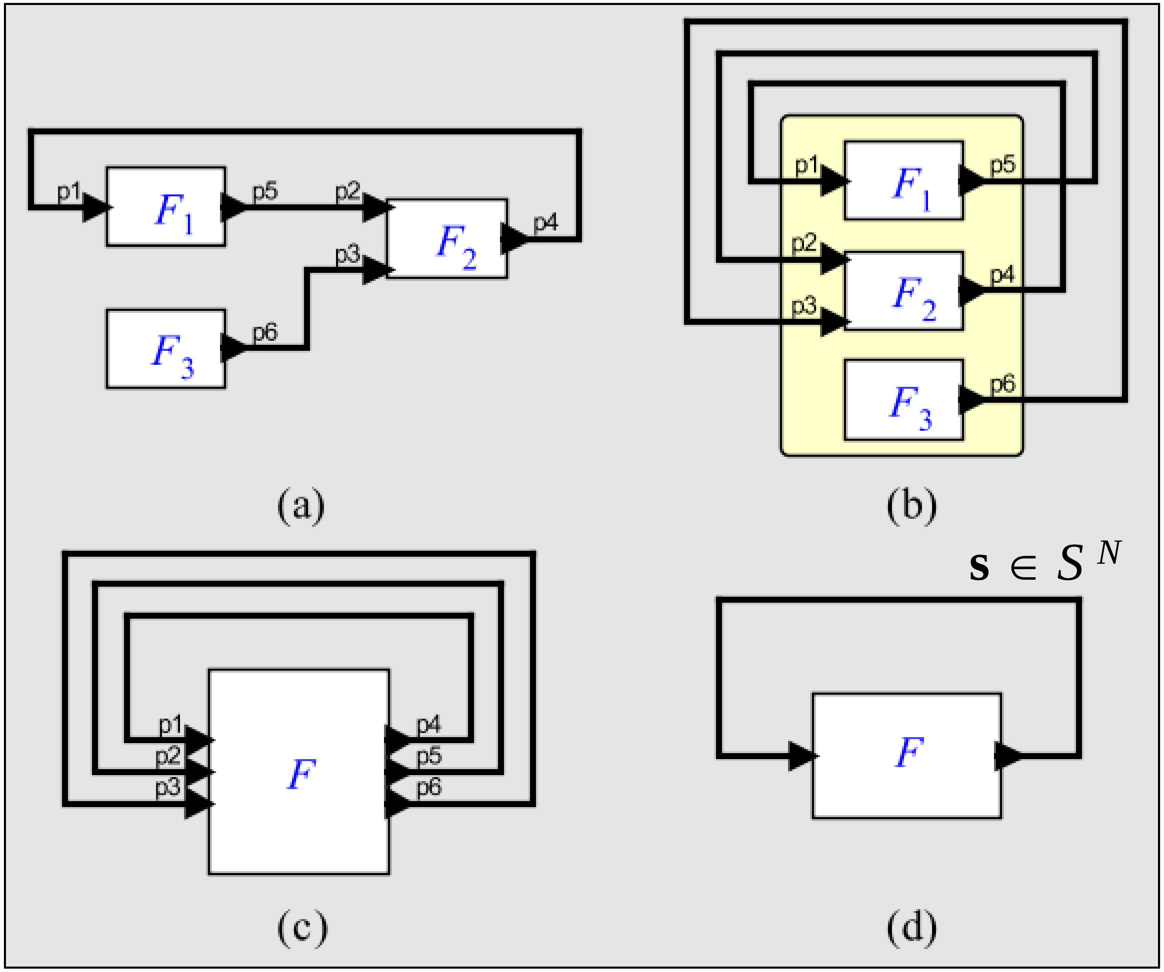


$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{1}{I_{yy}} \int_0^t T_y(\tau)\,d\tau$$

$$= \dot{\theta}_y(0) + \frac{K}{I_{yy}} \int_0^t (\psi(\tau) - \dot{\theta}_y(\tau))\,d\tau$$

Angular velocity appears on both sides. The semantics (meaning) of the model is the solution to this equation.

Observation: Any Composition is a feedback composition

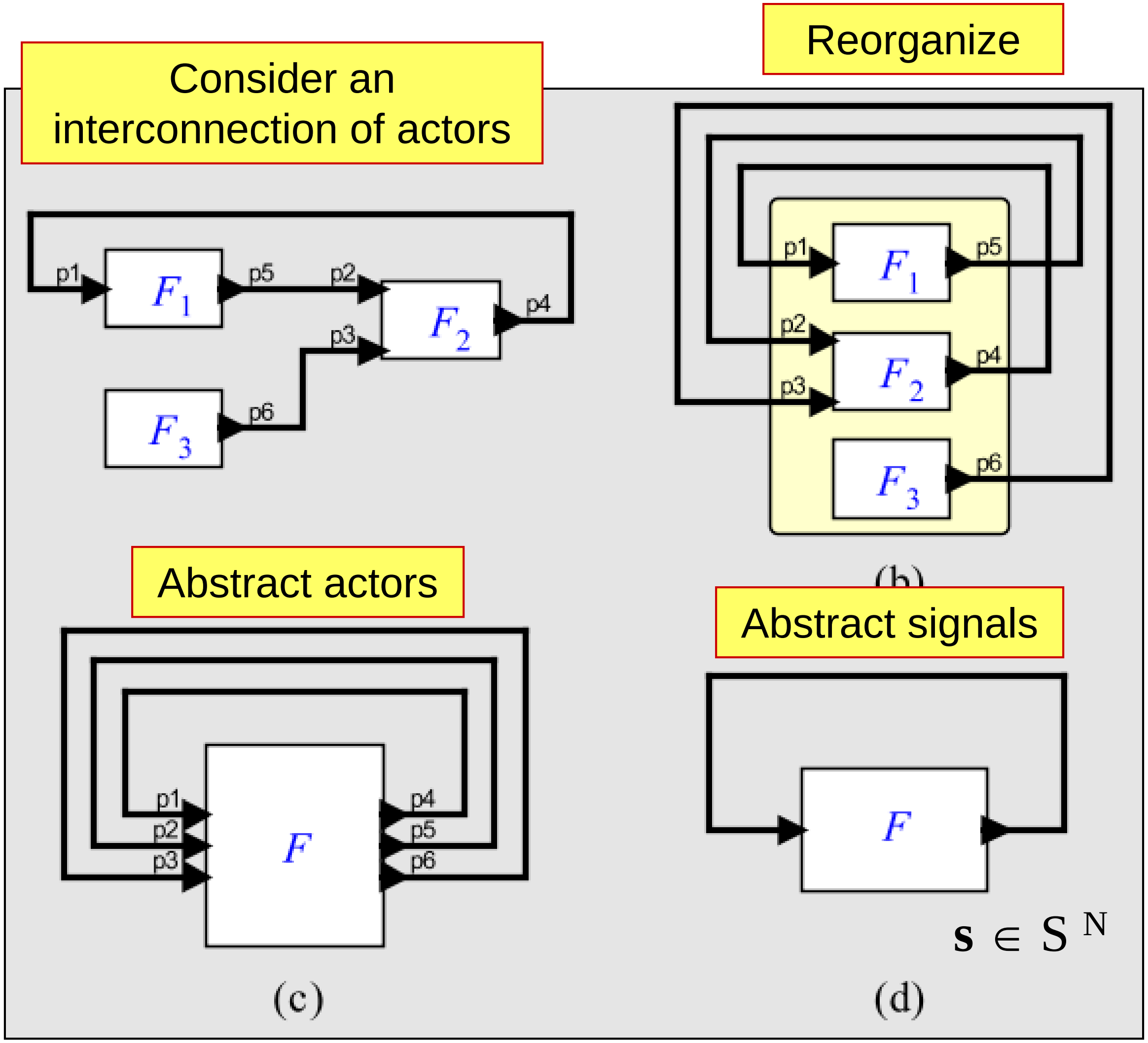


If every actor is a function then the semantics of the overall system is a

$\mathrm{s} \in S^N$ such that $F(\mathrm{s}) = \mathrm{s}$.

The behavior of the system is a "fixed point."

Fixed point semantics



We seek an $\mathbf{s} \in S^N$ that satisfies $F(\mathbf{s}) = \mathbf{s}$.

Such an **s** is called a *fixed point*.

We would like the fixed point to exist and be unique. And we would like a constructive procedure to find it.

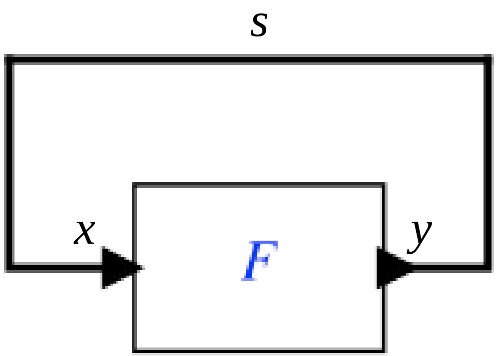
Data types

As with any connection, we require compatible data types:

$$V_y \subseteq V_x$$

Then the signal on the feedback loop is a function

$$s \colon \mathbb{N} \to V_y \cup \{absent\}$$



Then we seek s such that

$$F(s) = s$$

where F is the actor function, which for determinate systems has form

$$F \colon (\mathbb{N} \to V_x \cup \{absent\}) \to (\mathbb{N} \to V_y \cup \{absent\})$$

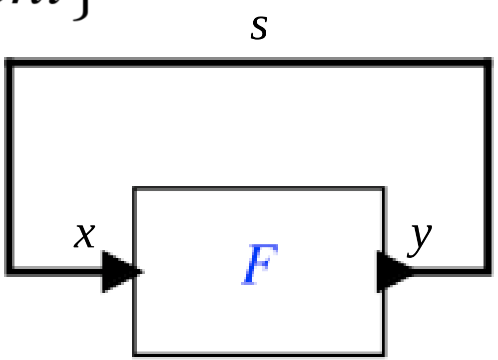
Firing functions

With synchronous composition of determinate state machines, we can break this down by reaction. At the n-th reaction, there is a (state-dependent) function

$$f(n) : V_x \cup \{absent\} \rightarrow V_y \cup \{absent\}$$

such that

$$s(n) = (f(n))(s(n))$$



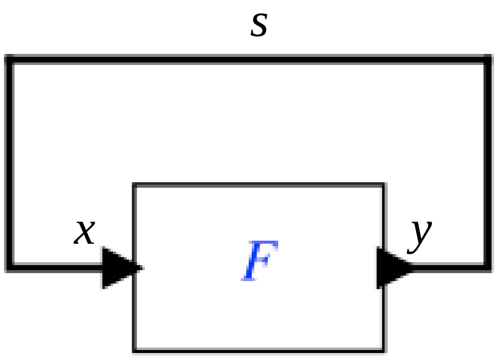
This too is a fixed point.

Well-formed feedback

At the n-th reaction, we seek $s(n) \in V_y \cup \{absent\}$ such that

$$s(n) = (f(n))(s(n))$$

There are two potential problems:

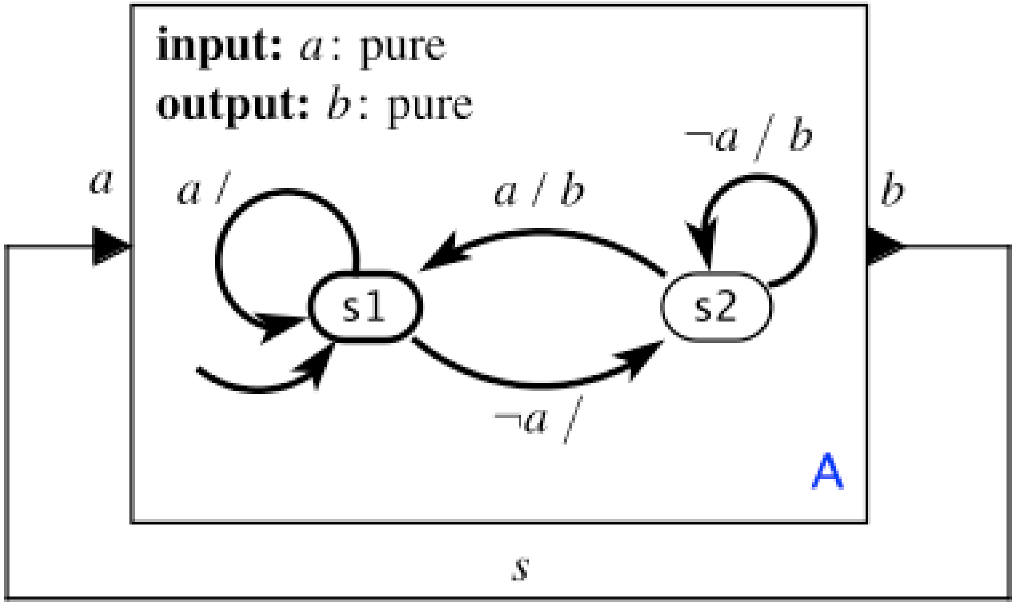


1. It does not exist.
2. It is not unique.

In either case, we call the system **ill formed**. Otherwise, it is **well formed**.

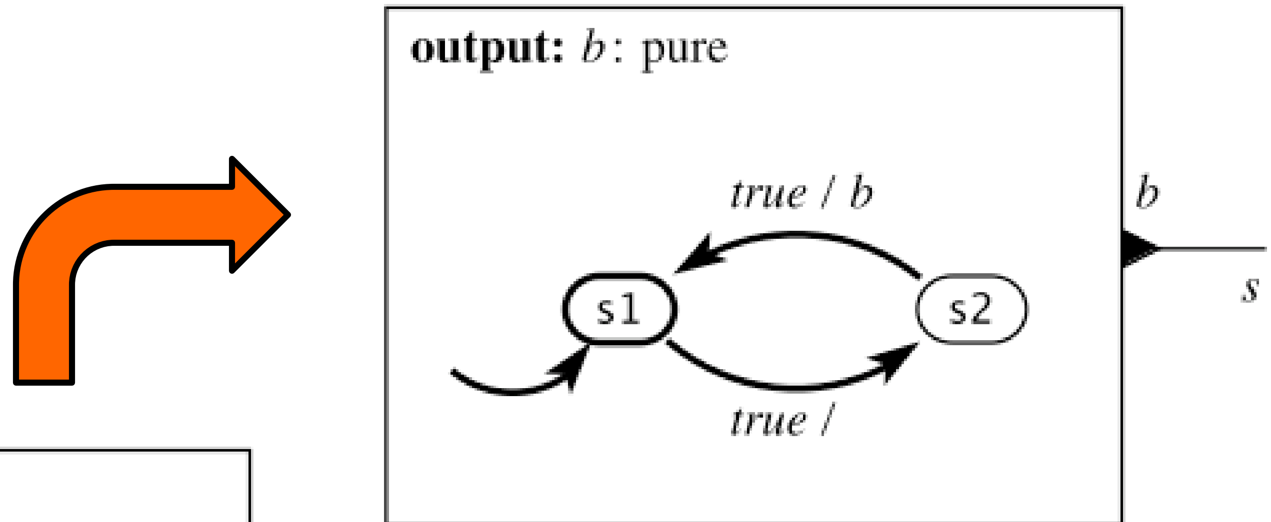
Note that if a state is not reachable, then it is irrelevant to determining whether the machine is well formed.

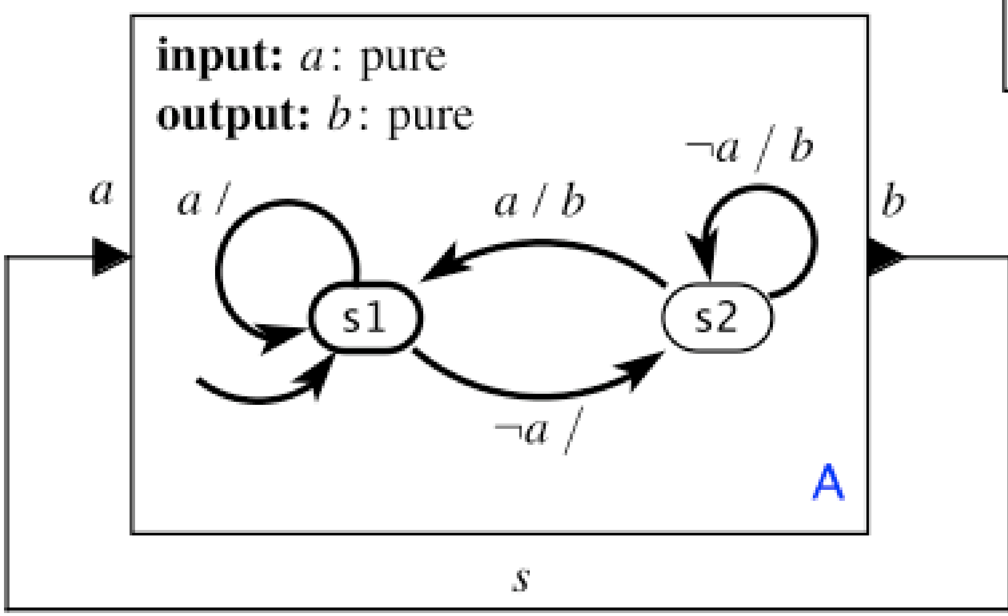
Well-formed example



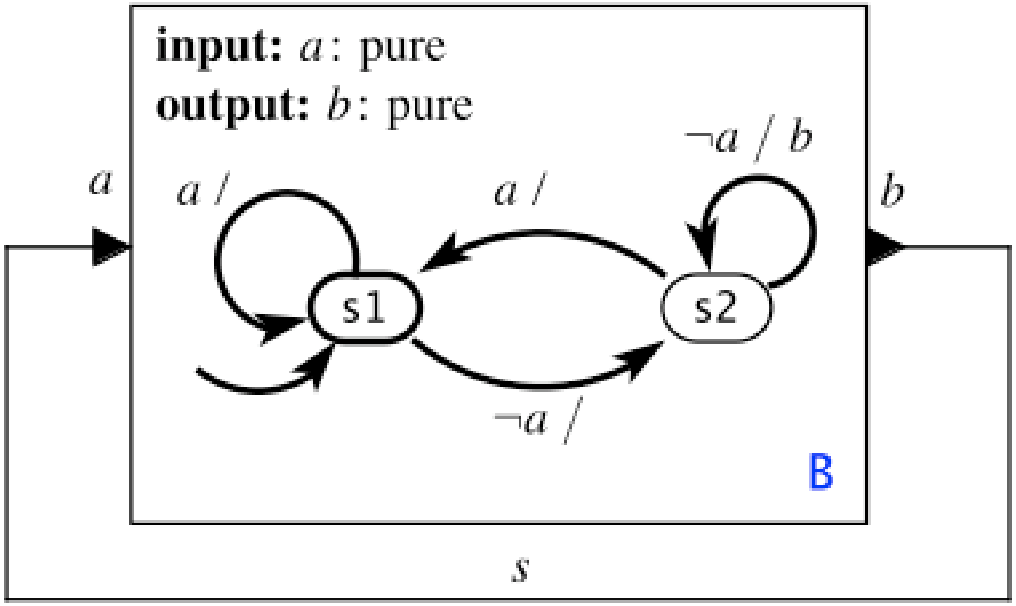
In state s1, we get the unique $s(n) = absent$.
In state s2, we get the unique $s(n) = present$.
Therefore, s alternates between *absent* and *present*.

Composite machine



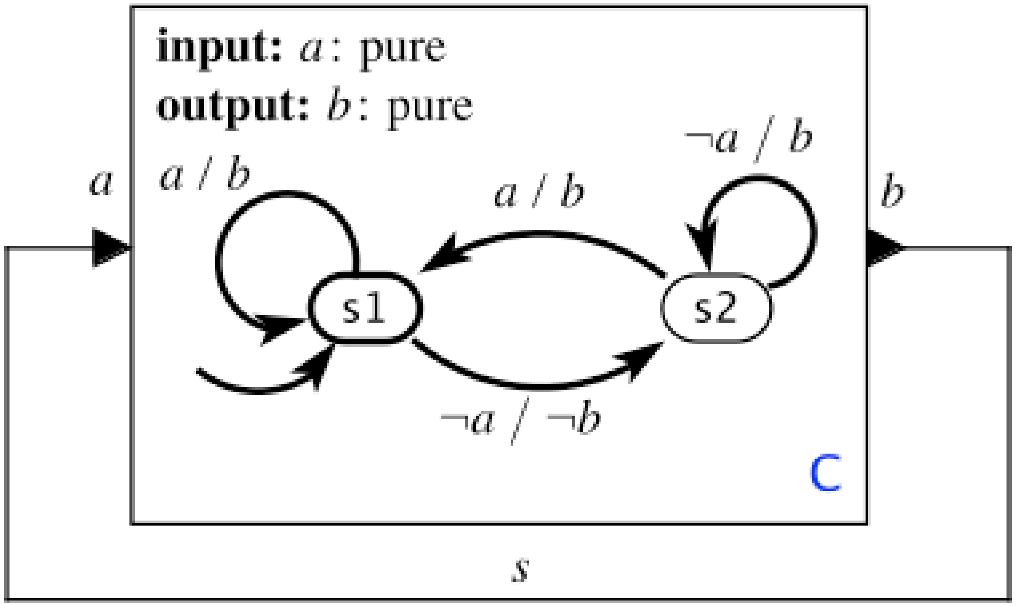


Ill-Formed Example 1 (Existence)



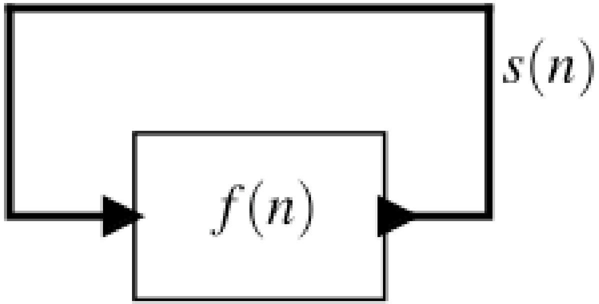
In state s1, we get the unique $s(n) = absent$.
In state s2, there is no fixed point.
Since state s2 is reachable, this composition is ill formed.

Ill-Formed Example 2 (Uniqueness)



In s1, both $s(n) = absent$ and $s(n) = present$ are fixed points.
In state s2, we get the unique $s(n) = present$.
Since state s1 is reachable, this composition is ill formed.

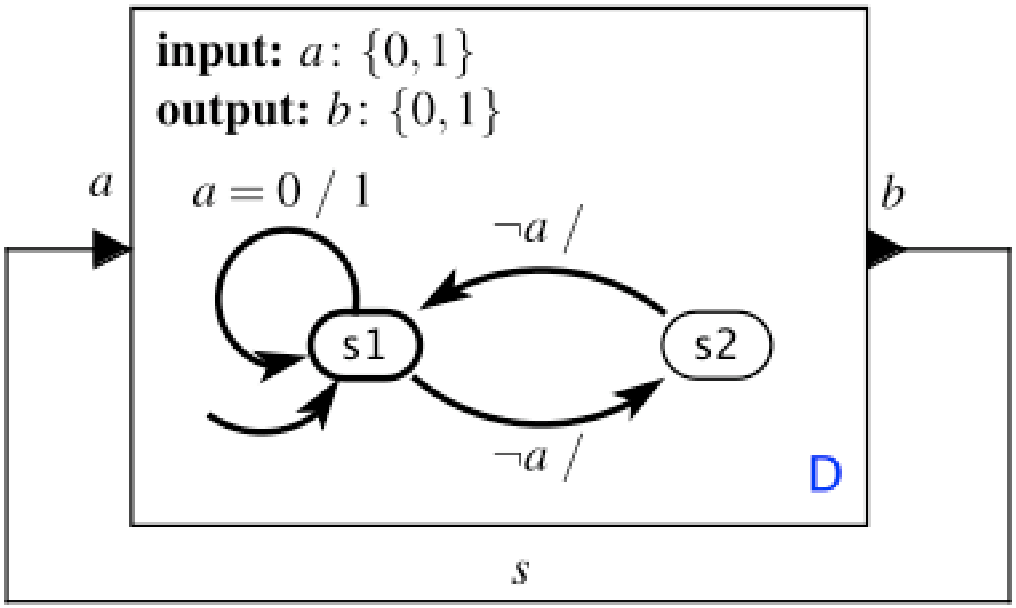
Constructive Semantics: Single Signal



1. Start with $s(n)$ *unknown*.
2. Determine as much as you can about $(f(n))(s(n))$.
3. If $s(n)$ becomes known (whether it is present, and if it is not pure, what its value is), then we have a unique fixed point.

A state machine for which this procedure works is said to be **constructive**.

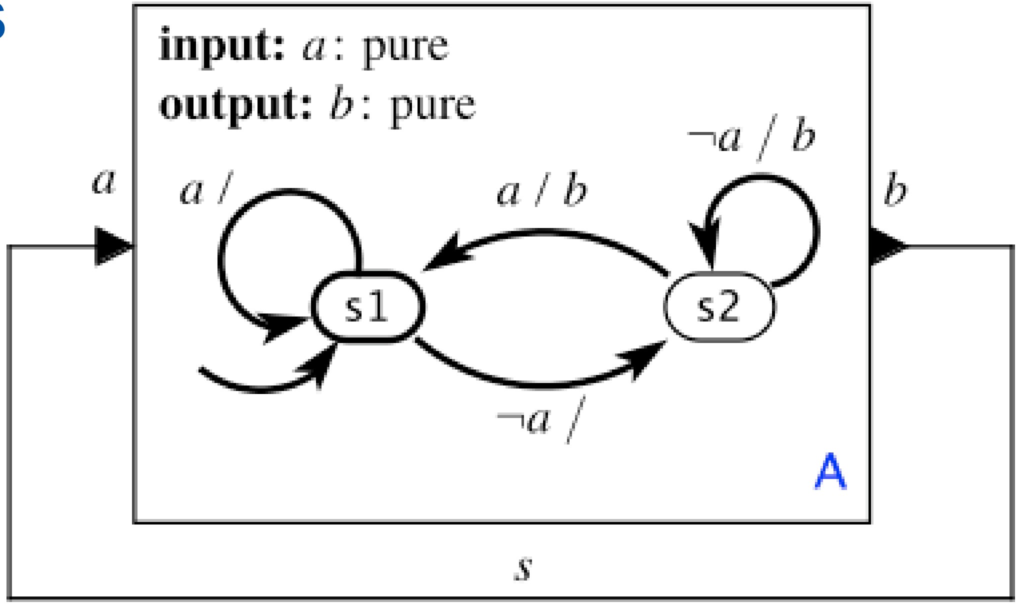
Non-Constructive Well-Formed State Machine



In state s1, if the input is unknown, we cannot immediately tell what the output will be. We have to try all the possible values for the input to determine that in fact $s(n) = absent$ for all n.

For non-constructive machines, we are forced to do **exhaustive search**. This is only possible if the data types are finite, and is only practical if the data types are small.

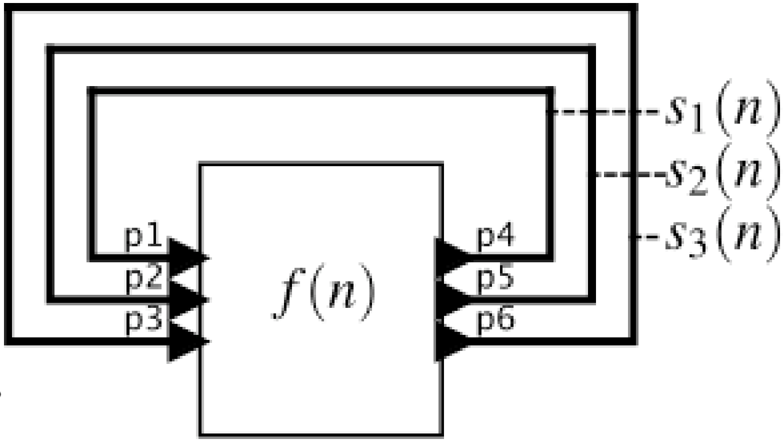
Must / May Analysis



For the above constructive machine, in state s1, we can immediately determine that the machine *may not* produce an output. Therefore, we can immediately conclude that the output is *absent*, even though the input is unknown.

In state s2, we can immediately determine that the machine *must* produce an output, so we can immediately conclude that the output is *present*.

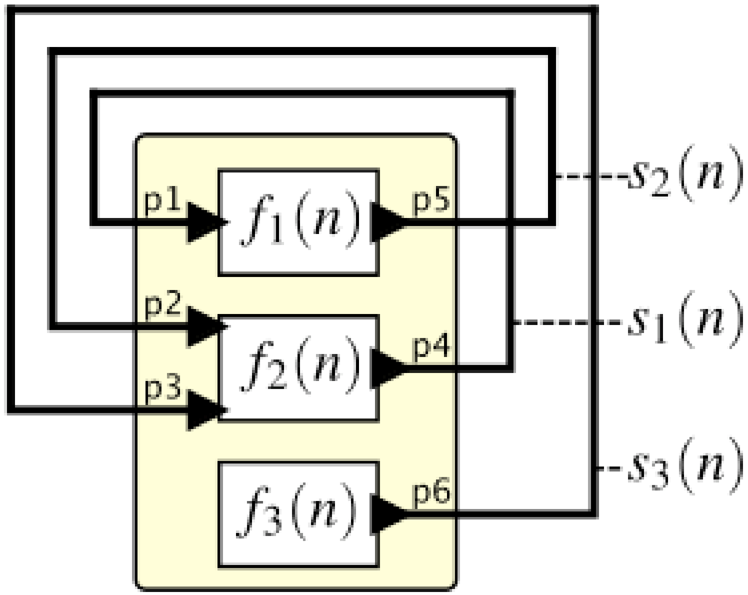
Constructive Semantics: Multiple Signals



1. Start with $s_1(n), \cdots, s_N(n)$ *unknown*.
2. Determine as much as you can about $(f(n))(s_1(n), \cdots, s_N(n))$.
3. Using new information about $s_1(n), \cdots, s_N(n)$, repeat step (2) until no information is obtained.
4. If $s_1(n), \cdots, s_N(n)$ all become known, then we have a unique fixed point and a constructive machine.

A state machine for which this procedure works is said to be **constructive**.

Constructive Semantics: Multiple Actors



- Procedure is the same.