State Machines Composition

Marco Di Natale

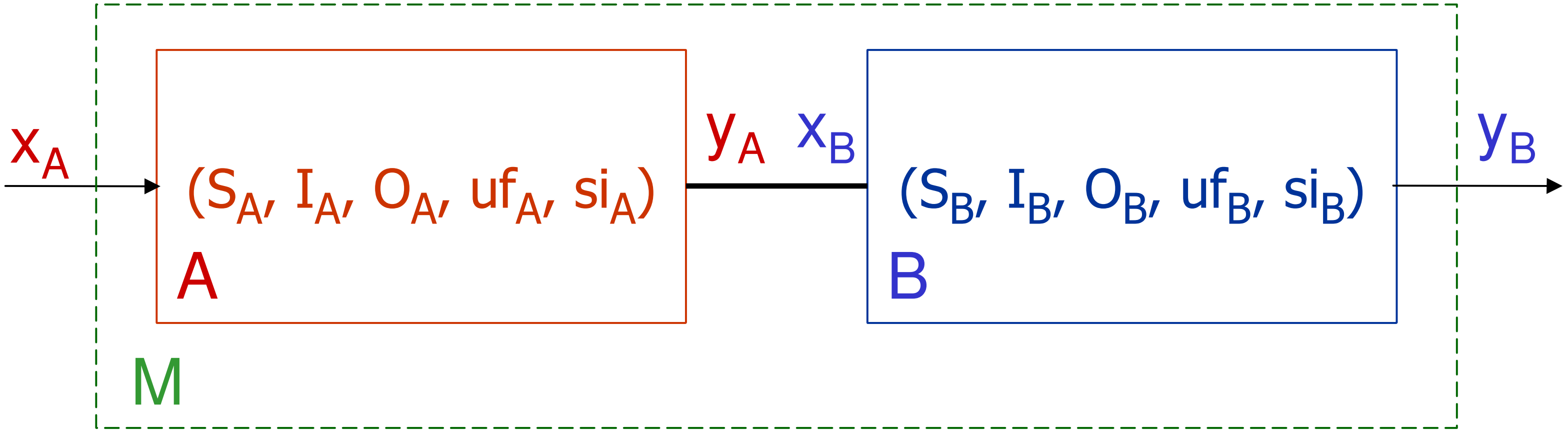
marco@sssup.it

Derived from original set by L. Palopoli @ Uni Trento

Synchrony

- Each state machine in the composition reacts to external inputs *simultaneously* and *instantaneously*
- Our system react to external stimuli: for this reason they are called *reactive*
- Because our systems react synchronously to external inputs they are called synchronous/reactive (SR)

Series (Cascade) Composition



Here, the output of machine A is the input of machine B.

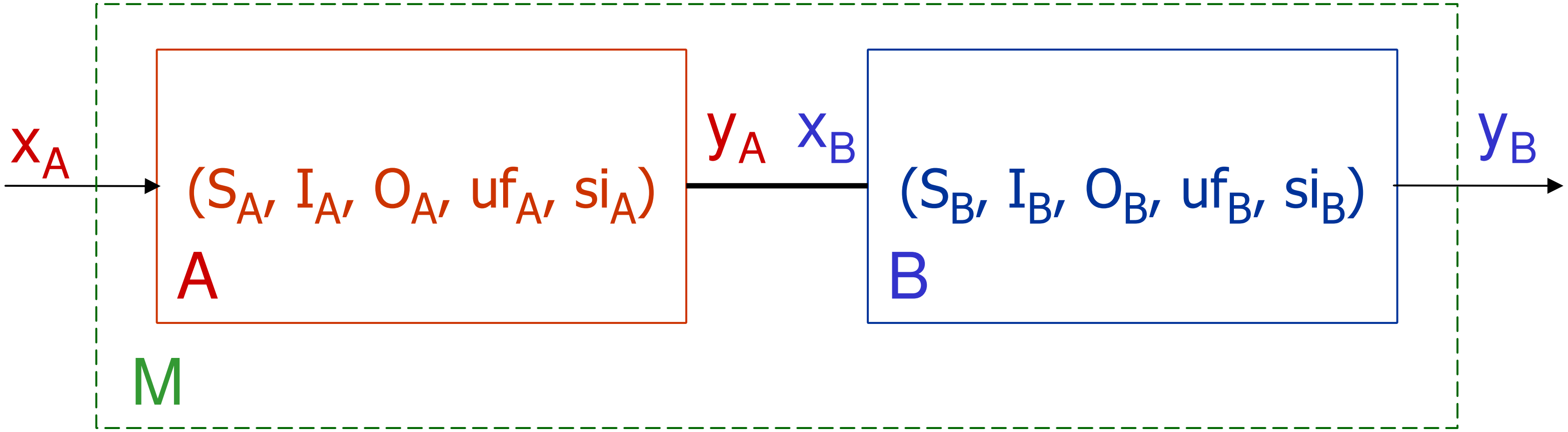
The two machines react simultaneously (both on step n)

Each machine has its own input, current state, and output.

The effect of the input $x_A(n)$ propagates instantaneously through the cascade at each step: **synchrony**

We may view the cascade of two machines as a single machine.

Series Composition: Definition



Define the 5-uple for the composite machine:

Inputs = *Inputs*$_A$

Outputs = *Outputs*$_B$

States = *States*$_A$ x *States*$_B$

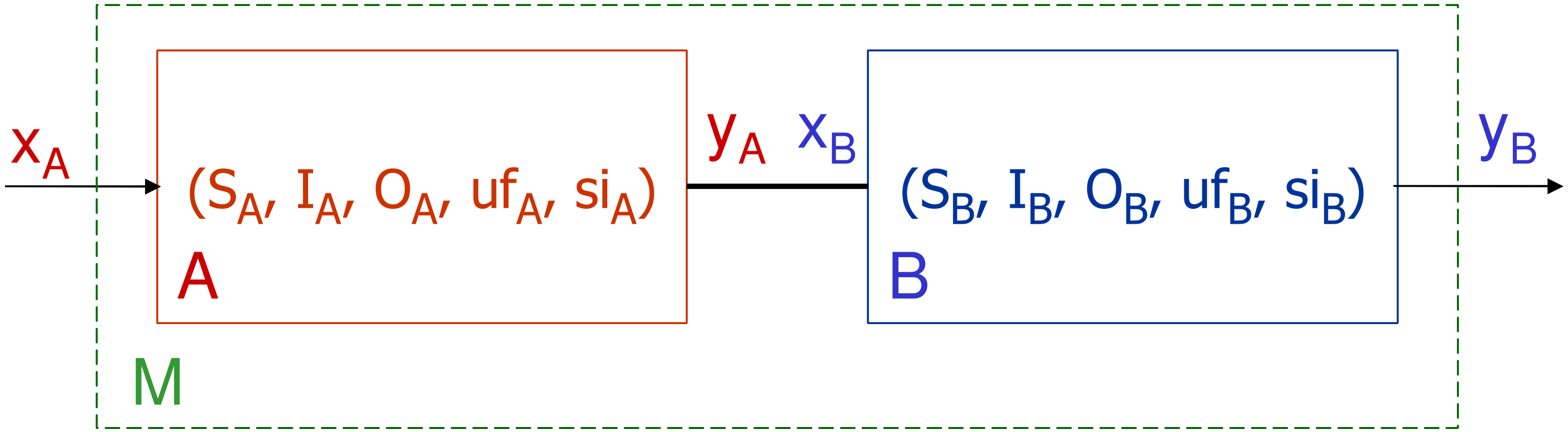
initialState = (*initialState*$_A$, *initialState*$_B$)

update($(s_A(n), s_B(n)), x(n)$) = ($(s_A(n+1), s_B(n+1)), y(n)$)

where $(s_A(n+1), y_A(n))$ = *update*$_A$$(s_A(n), x(n))$

and $(s_B(n+1), y(n))$ = *update*$_B$$(s_B(n), y_A(n))$

Series Composition: Interconnection



Note that the "internal" output $y_A(n)$ is used as the "internal" input $x_B(n)$ to machine B.

Thus, for the cascade connection to be valid, we must have

$$Outputs_A \subset Inputs_B$$

Series Composition: Diagram

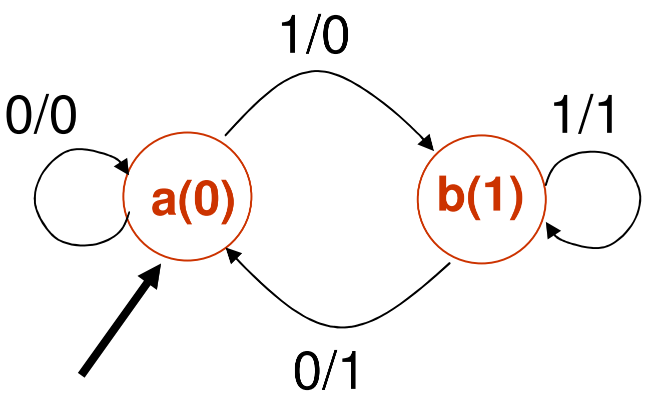
The state diagram for the series composition is computed by the following algorithm:

1. Draw a circle for each state in $States_A$ x $States_B$.
2. For each state, consider each possible input to machine A.
   a) Find the corresponding next state in machine A.
   b) Find the output of machine A, which forms the input of machine B.
   c) Find the corresponding next state in machine B.
   d) Find the output of machine B.
   e) Draw the transition arrow to $(s_A(n+1), s_B(n+1))$.
   f) Label the transition arrow with the input to machine A and the output from machine B.

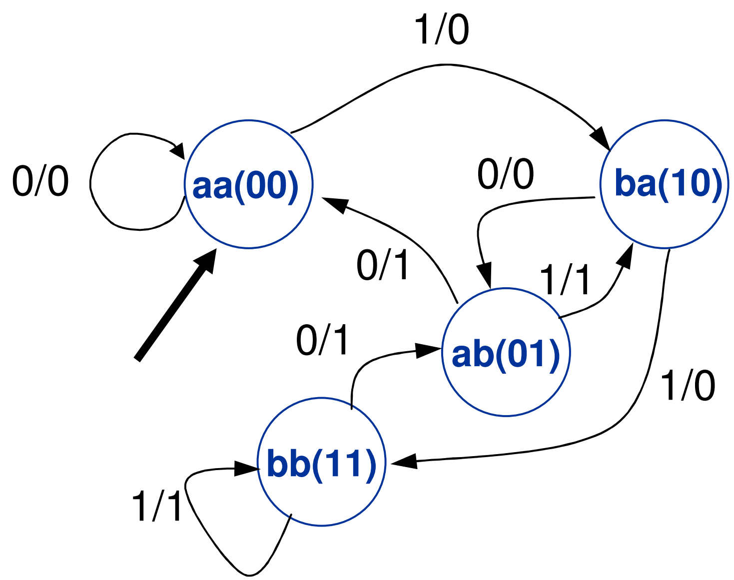
Series Composition: Example1

Consider the cascade of machine A (1-unit delay) with itself

Find the composite state response and output for input x = 1 0 0 1.



Machine A



Machine A→A

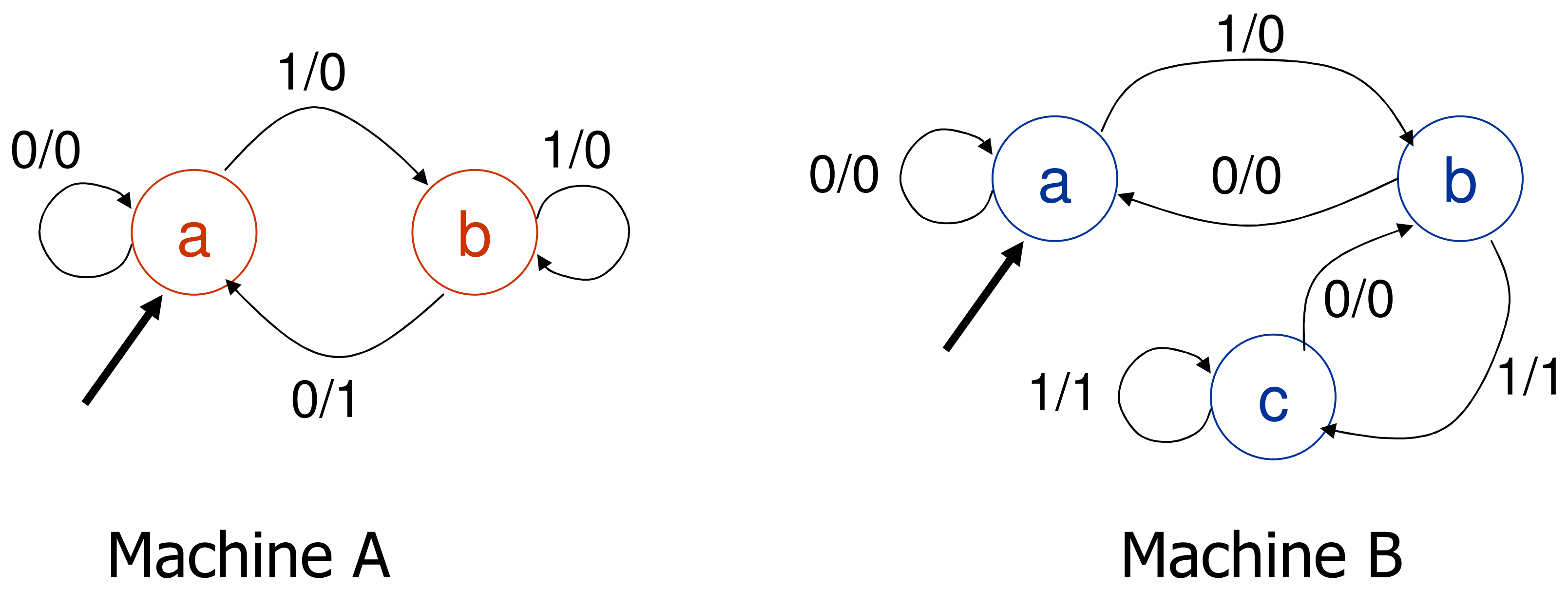
s = (a, a), (b, a), (a, b), (a, a), (b, a)

y = 0 0 1 0

Series Composition: Example2

Consider the cascade of machine A and machine B below, where the output of machine A is the input of machine B.

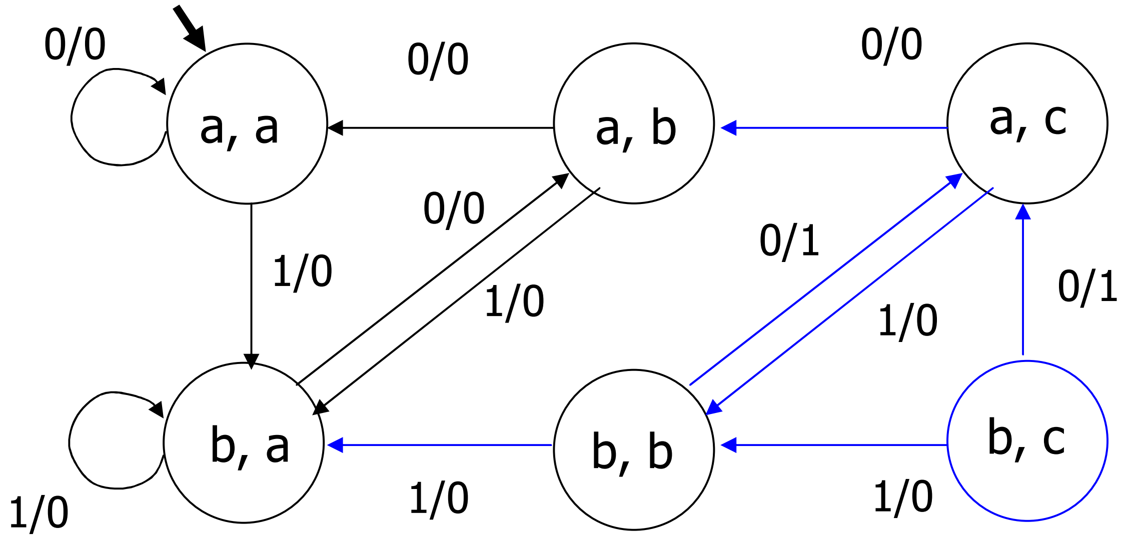
Find the composite state response and output for input x = 1 0 0 .



s = (a, a), (b, a), (a, b), (a, a)   y = 0 0 0

Series Composition: Diagram

Try drawing a single state diagram for machines A and B in the previous example:

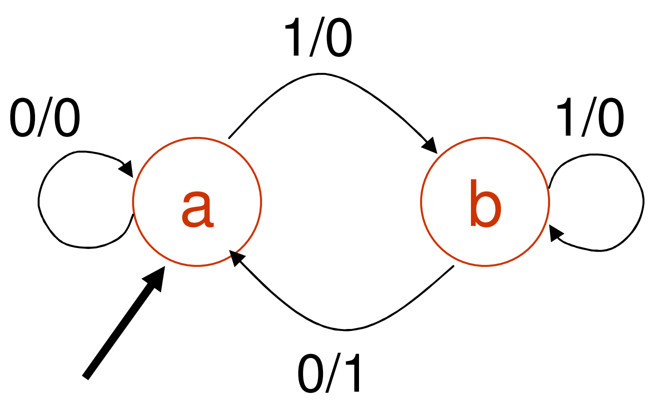


Can we ever get to state (b, c)?

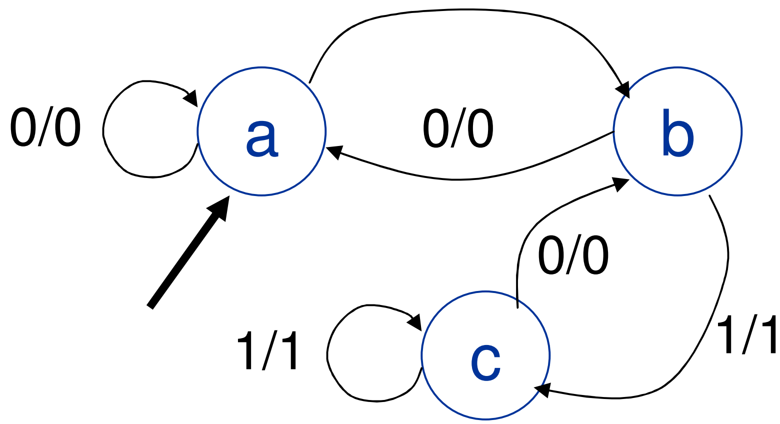
Can we ever get to states (a, c) or (b, b)?

Can we ever get a nonzero output?

Reachability



Machine A



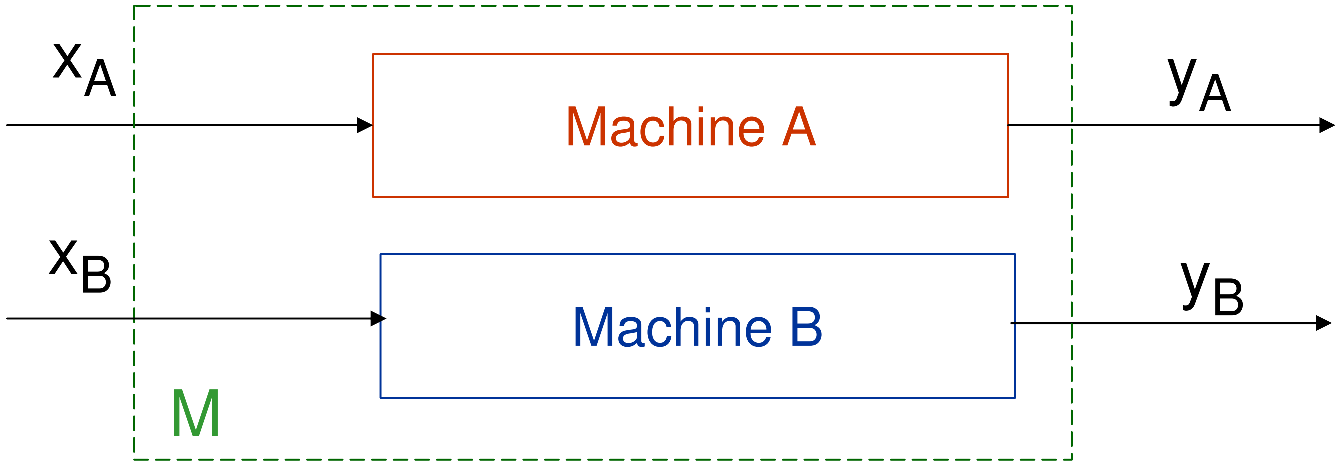
Machine B

On its own, given its entire set of legal inputs, Machine B can reach state c and give an output of 1.

But, in cascade, the inputs of Machine B are limited to the possible outputs of Machine A.

Machine A cannot generate a sequence of outputs that would drive machine B into state c. This behavior is not in Behaviors_A.

Parallel Connection



States = *StatesA* x *StatesB*

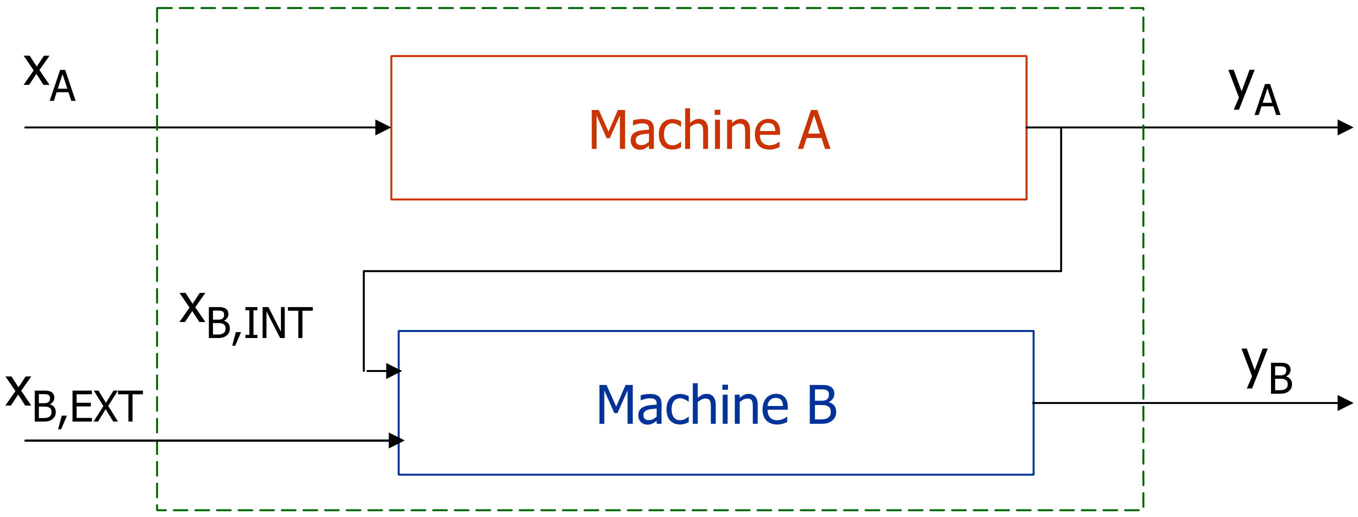
initialState = (*initialStateA*, *initialStateB*)

Output=(y_A x y_B) *Intput*=(x_A x x_B)

update(($s_A(n)$, $s_B(n)$), $x(n)$) = (($s_A(n+1)$, $s_B(n+1)$), ($y_A(n)$,$y_B(n)$))

- where ($s_A(n+1)$, $y_A(n)$) = *updateA*($s_A(n)$, $x_A(n)$)
- and ($s_B(n+1)$, $y_B(n)$) = *updateB*($s_B(n)$, $y_B(n)$)

More Complicated Connections

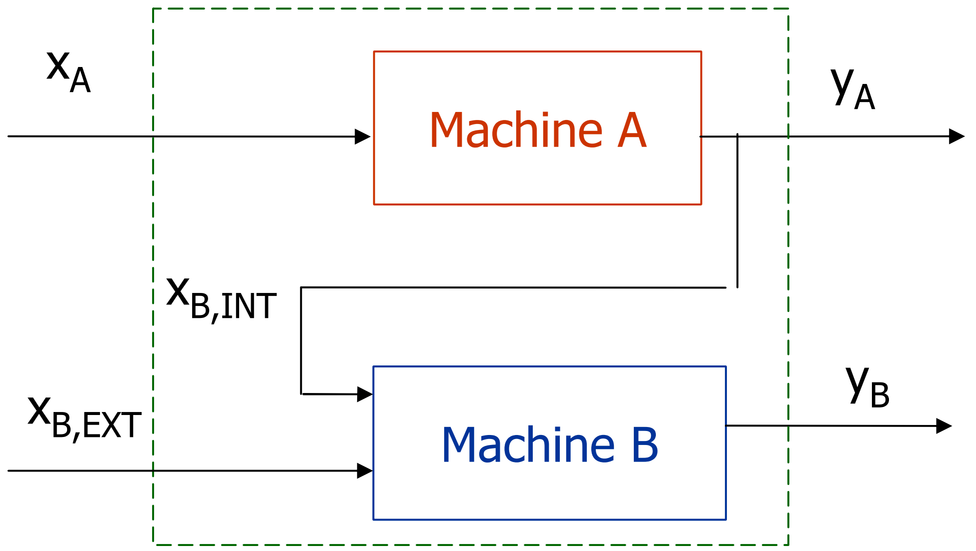


Here, we wish to have access to the individual output y_A, even when treating the cascade as one big machine.

Sending a signal to more than one destination is called **forking**.

Note also that there is an additional external input into machine B.

More Complicated Connections



Each arrow coming into or out of the composite machine originates from an interconnection point or **port**.

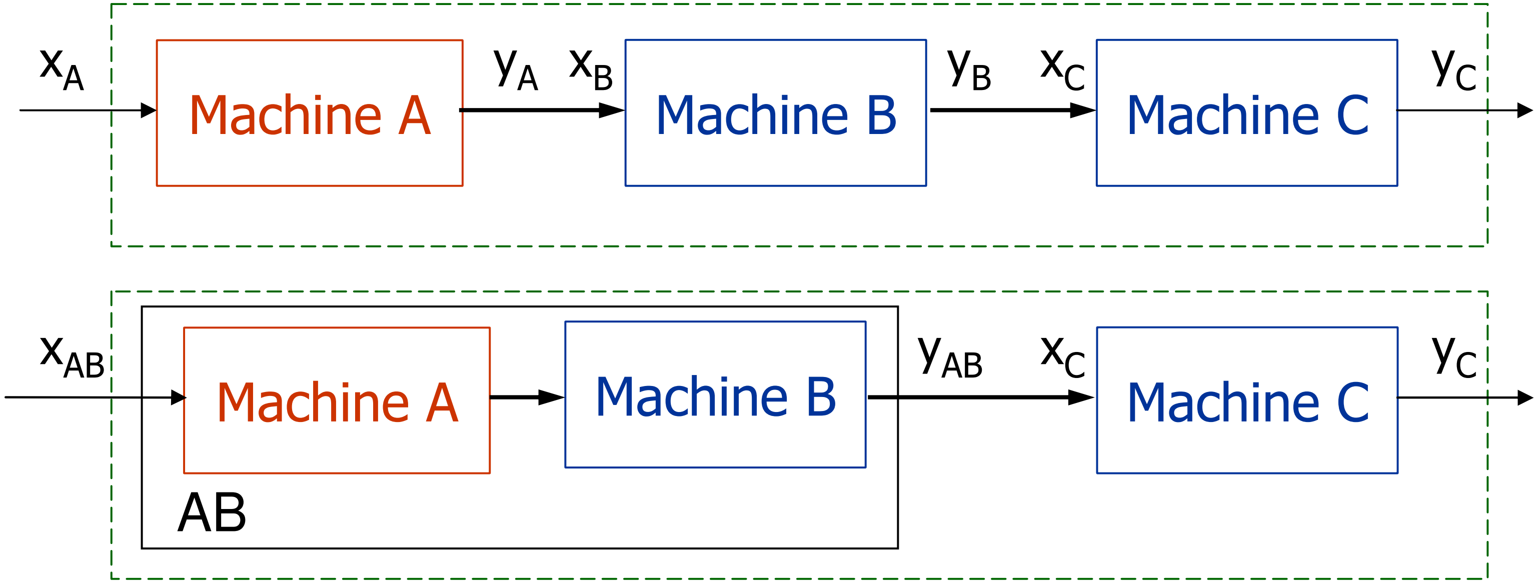
Here, there are two input ports and two output ports.

In the composite machine, we can express the input and output in the expected way using a set product:

$Inputs = Inputs_A \times Inputs_{B,EXT}$ $\quad$ $Outputs = Outputs_A \times Outputs_B$

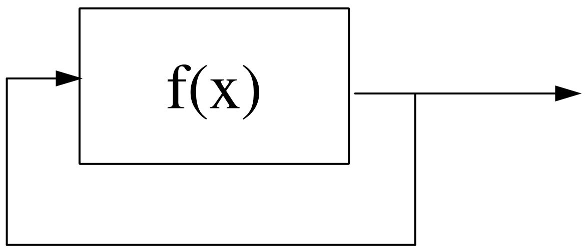
The set of inputs or outputs for a particular port (Outputs_B, for example) is called a **port alphabet**.

Hierarchical composition

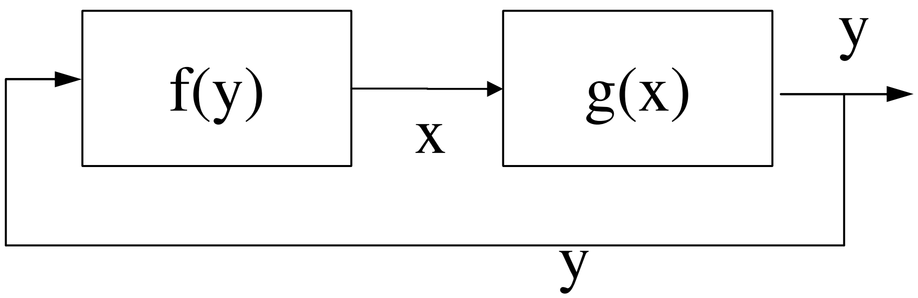


We can compose A and B and then C, or we can make it in the order A (BC) obtaining bisimilar machines (machines with the same behavior).

Feedback/Loop composition = Fixed point



Fixed point: x = f(x)



Fixed point: x = f(y), y = g(x)

Composing y = g(f(y))

Solutions of fixed point formula

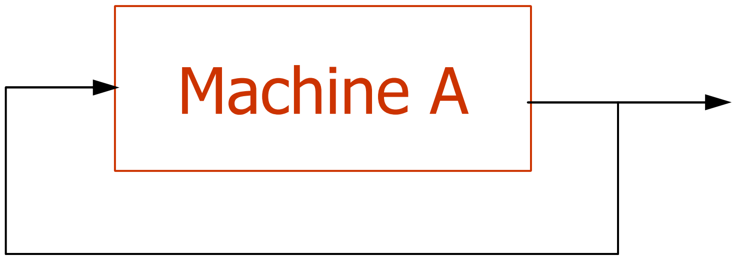
Problem: the result of the composition is no more a "system", the way we defined it.

The output update is no more a function. It is not guaranteed to have a solution.

Examples:

$f(x)=x^2$	$x=x^2$	$x=0, x=1$	2 solutions: ill-posed
$f(x)=x^2+1$	$x=x^2+1$	$x=\{\}$	no solutions: ill-posed
$f(x)=-x+1$	$x=-x+1$	$x=\frac{1}{2}$	well-posed

Particular case: sm with no external inputs



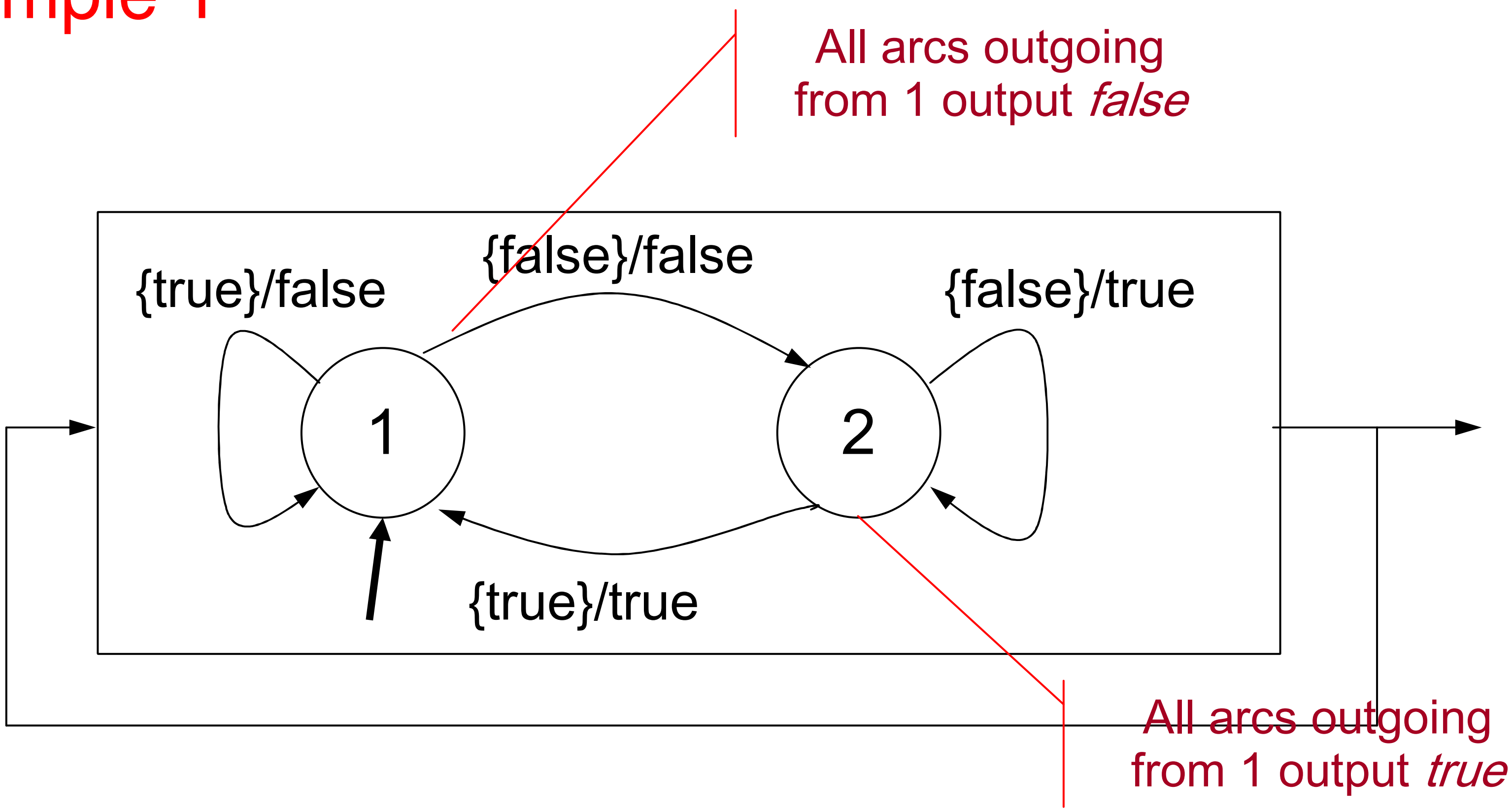
- Introduce two fictitious input symbols: *react* and *absent*

$$Outputs_A \subseteq Inputs_A$$
$$(s(n+1), y(n)) = Update_A(s(n), y(n))$$
$$y(n) = Output_A(s(n), y(n))$$

Solving the FP problem is the key point

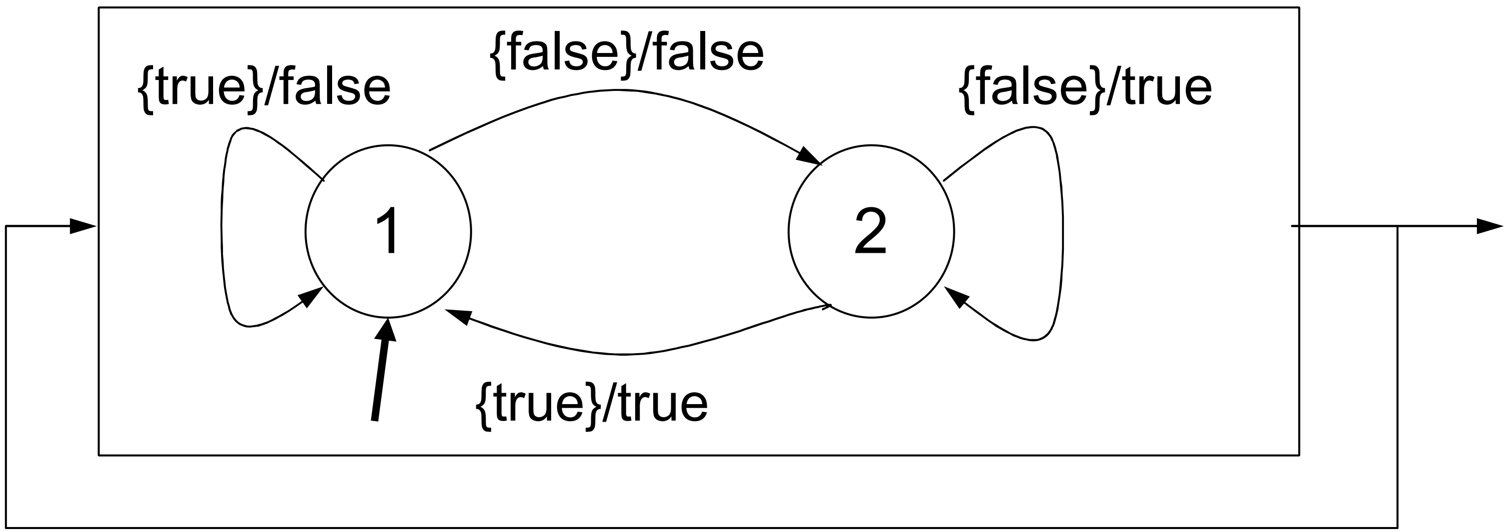
- Clearly the machine stuttering always works!
- We are interested in non-stuttering FP

Example 1



$$Outputs_A = Inputs_A = \{true, false, absent\}$$
$$y(n) = output_A(s(n), y(n))$$

Example 1



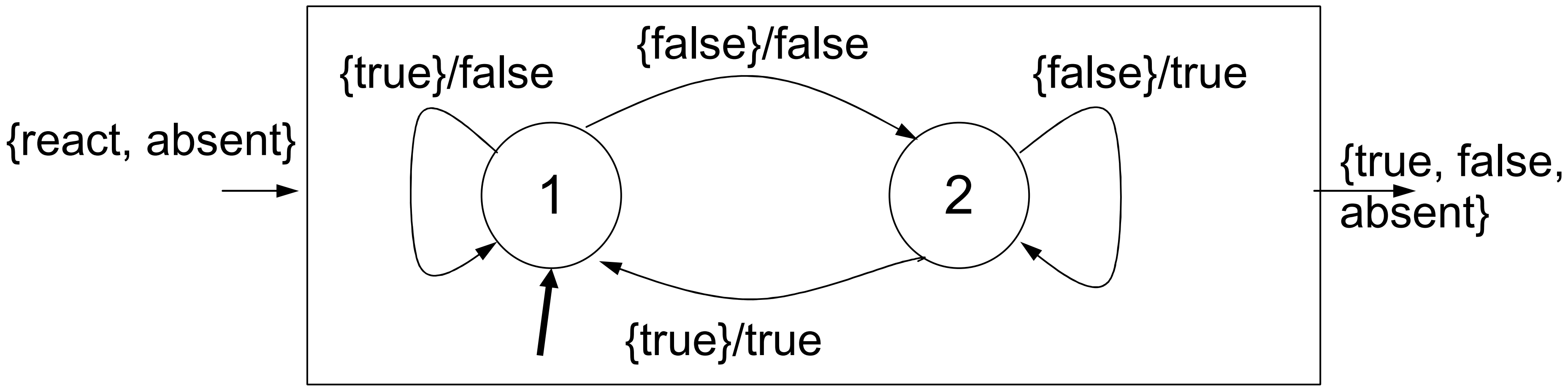
Given the state and the output we see if there is a fixed point solution

The state determines the output

$$false = output_A(1, false)$$

$$true = output_A(2, true)$$

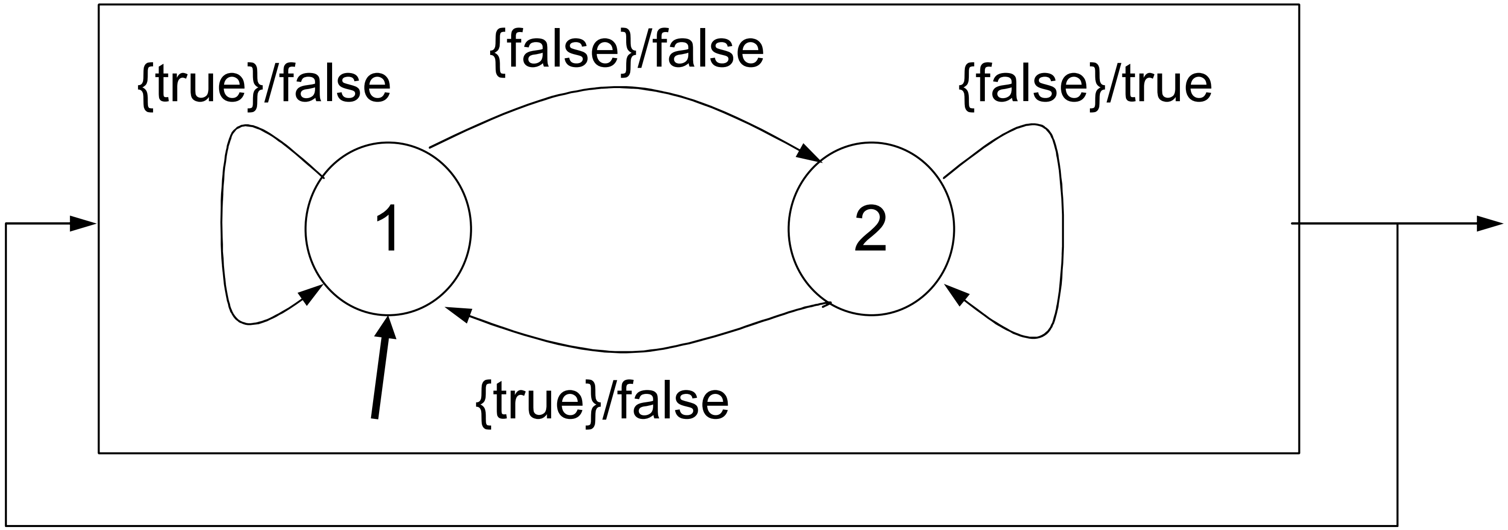
Example - Equivalent machine



$$false = output_A(1, false)$$
$$true = output_A(2, true)$$

$$\{react, react, react, react, \ldots\} \rightarrow \{false, true, false, true, \ldots\}$$

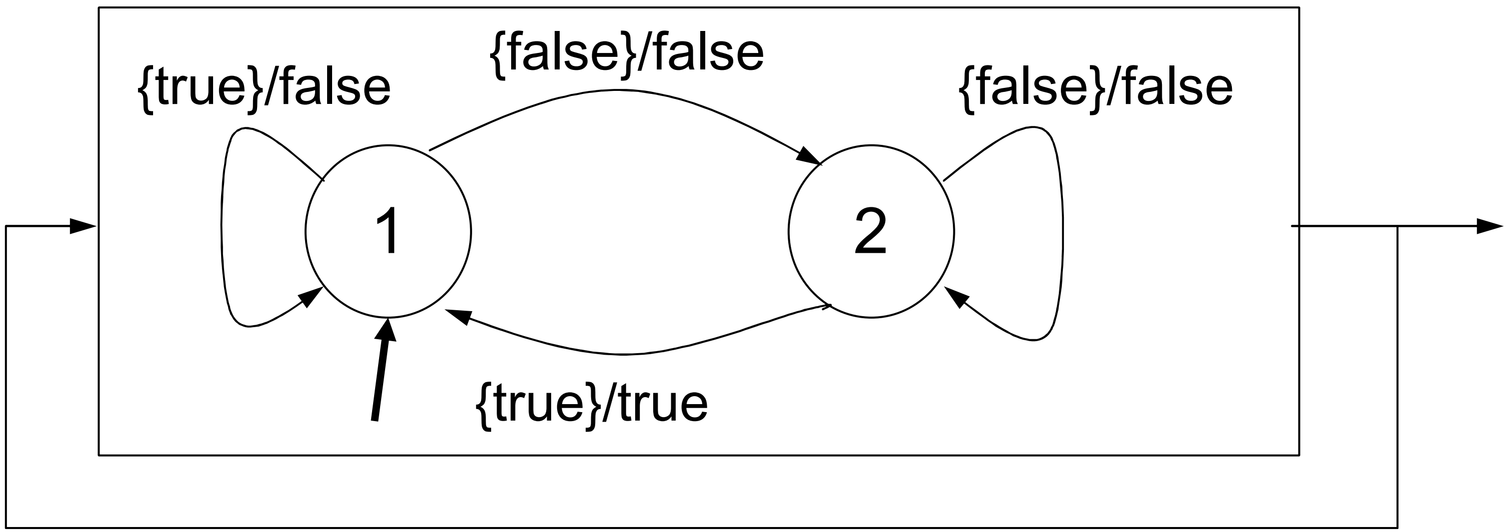
Example 2



There is no fp
solution for state 2

$$false = output_A(1, false)$$
$$y = output_A(2, y)$$

Example 3



There are two fp solutions for state 2

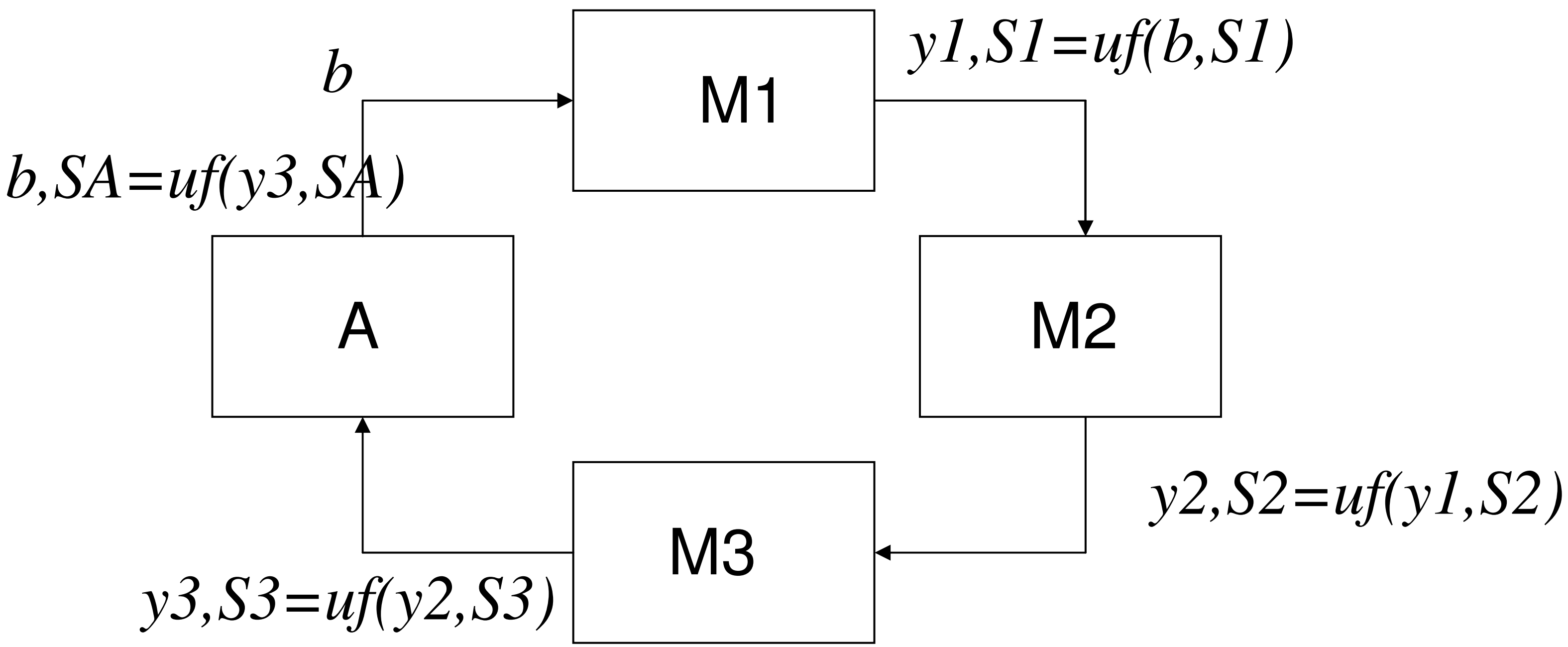
$$false = output_A(1, false)$$

$$y = output_A(2, y)$$

Sufficient condition for well-formedness

- Feedback composition is well formed if for each loop there is at least one FSM (A) for which the output is state-determined (all arcs outgoing from a state have the same output):

$$\forall reachable\ s_A(n),\ \forall\ x(n) \neq absent,\ update_A(s_A(n),\ x(n)) = b$$



State-determined outputs

- Feedback composition is well formed if the output is state-determined (all arcs outgoing from a state have the same output):

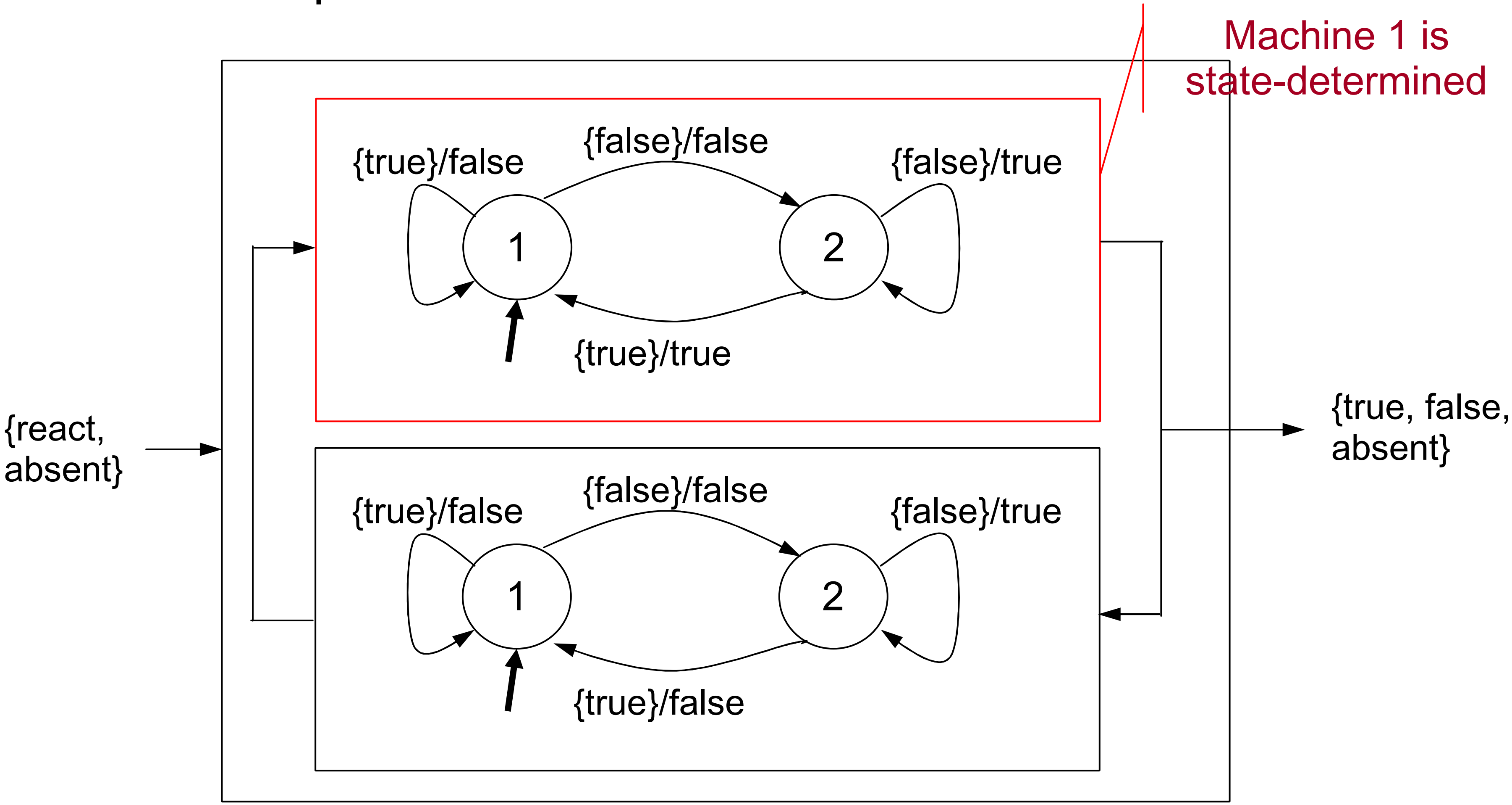
  $\forall reachable\ s(n),\ \forall\ x(n) \neq absent,\ update_A(s(n),\ x(n)) = b$

- In this case the composition is defined as follows:

$$states = states_A$$
$$Inputs = \{react, absent\}$$
$$Outputs = Outputs_A$$
$$initialState = intialState_A$$
$$update(s(n), x(n)) = \begin{cases} update_A(s(n), b)\ where\ b = output_A(s(n), x(n)) & if\ x(n) = react \\ (s(n), x(n)) & if\ x(n) = absent \end{cases}$$

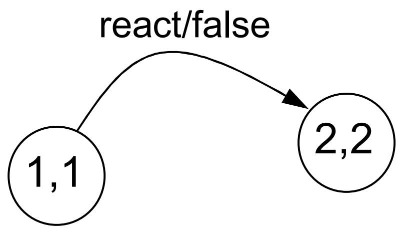
Example

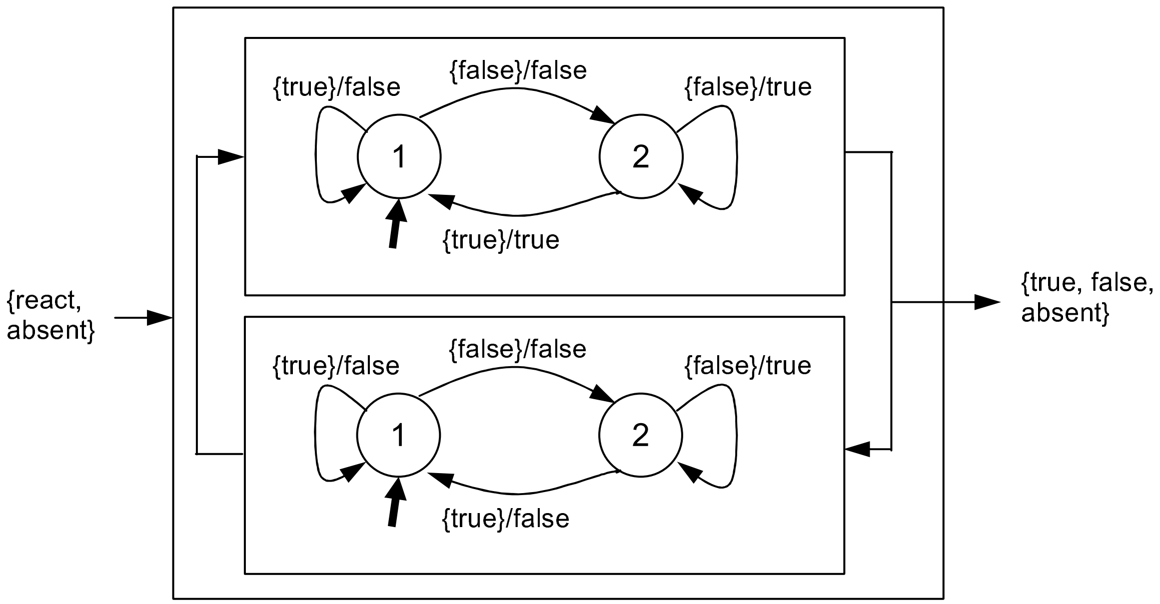
- State-determined outputs ensure well-formedness also in more complex situations



Example - (continued)

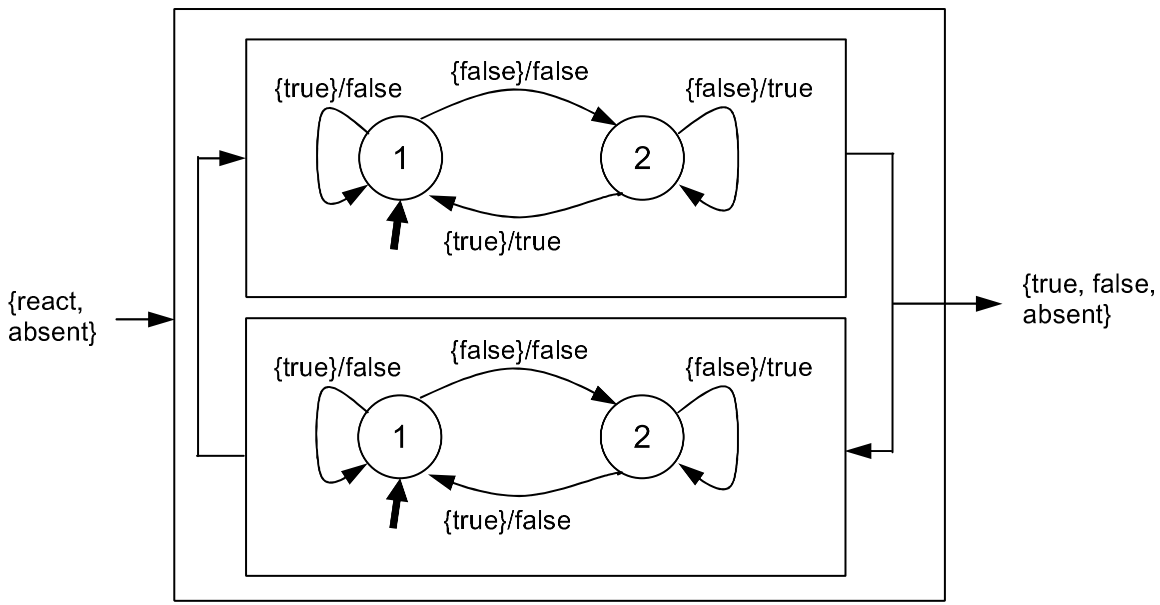
- Suppose both machines are in their initial states
  - The first emits false, which is propagated by the second:
    - Both go to 2

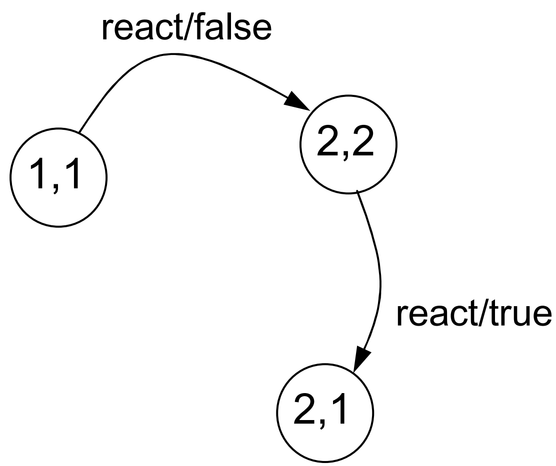




Example - (continued)

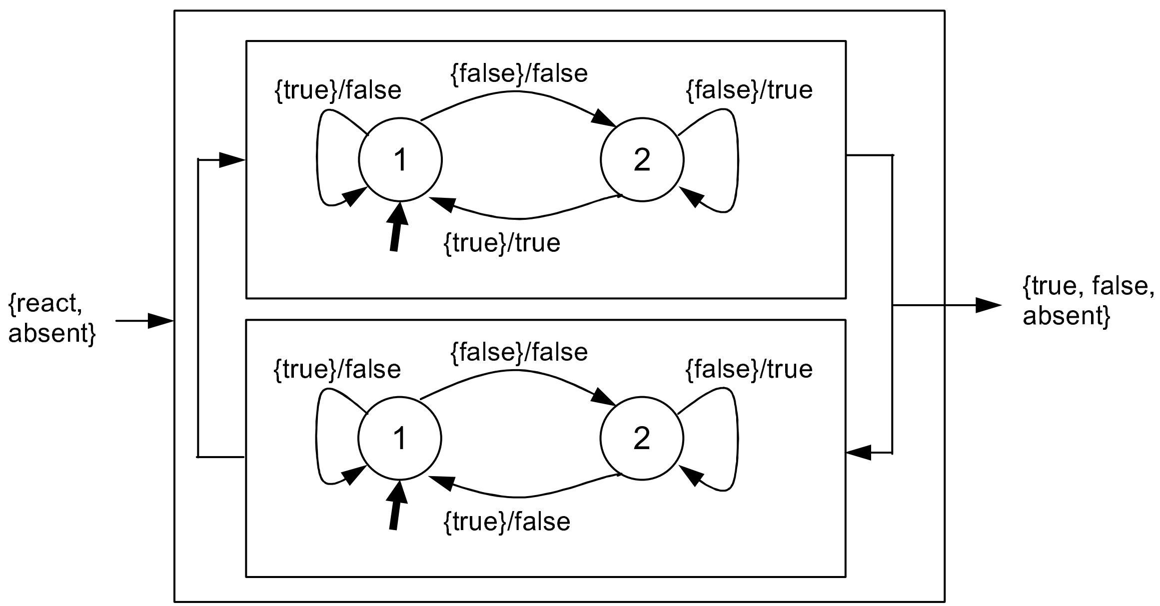
- Now, suppose both are in 2:
  - The top one emits true and the bottom one emits false
    - The top 1 remains in 2 and the bottom one goes to 1

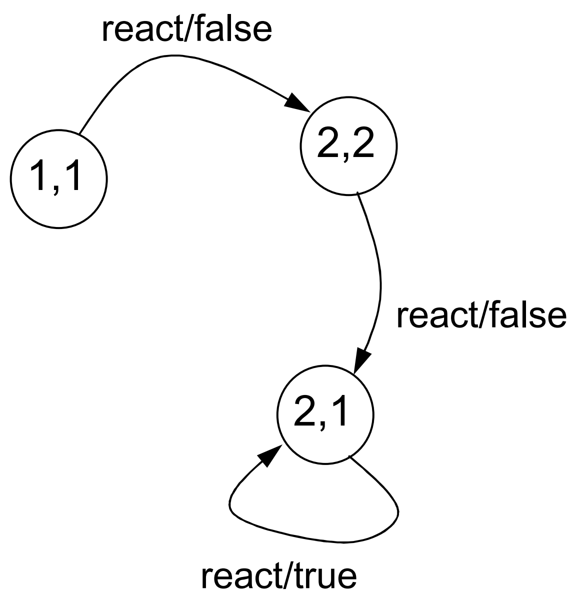




Example - (continued)

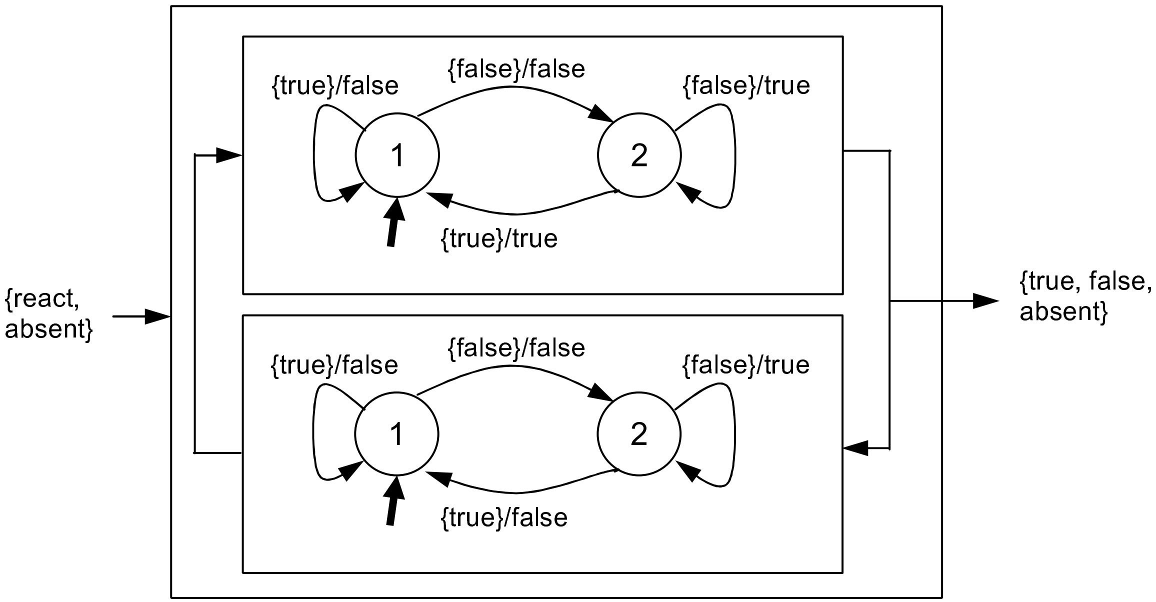
- Now, suppose top machine is in 2 and the bottom be in 1:
  - The top one emits true and the bottom one emits false
    - The top 1 remains in 2 and the bottom one remains in 1

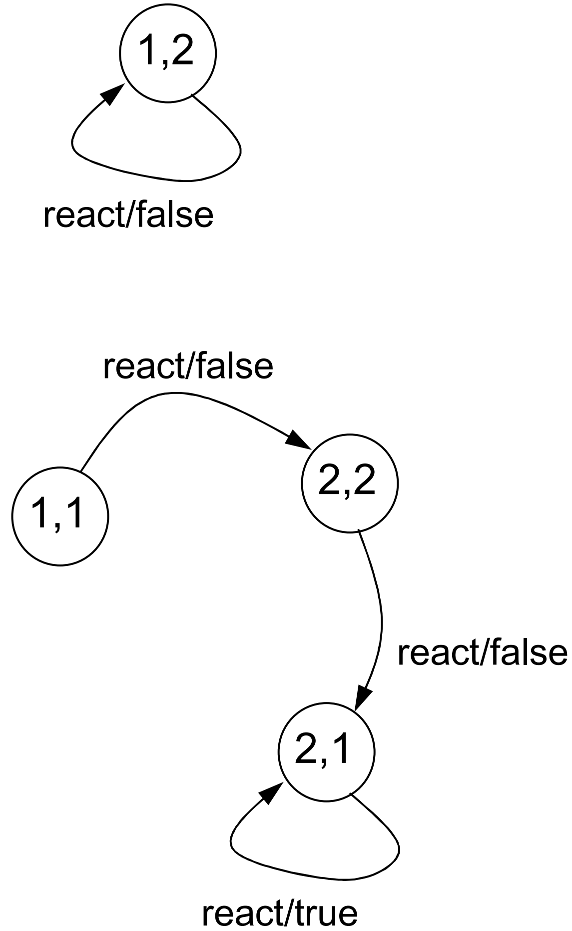




Example - (continued)

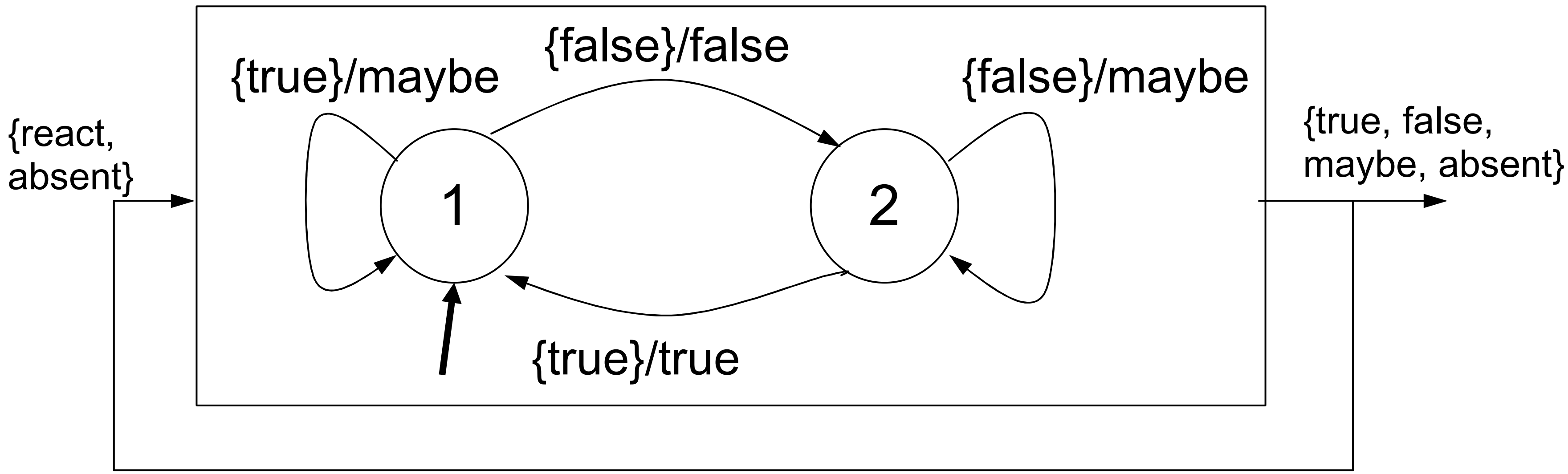
- state (1,2) is unreachable



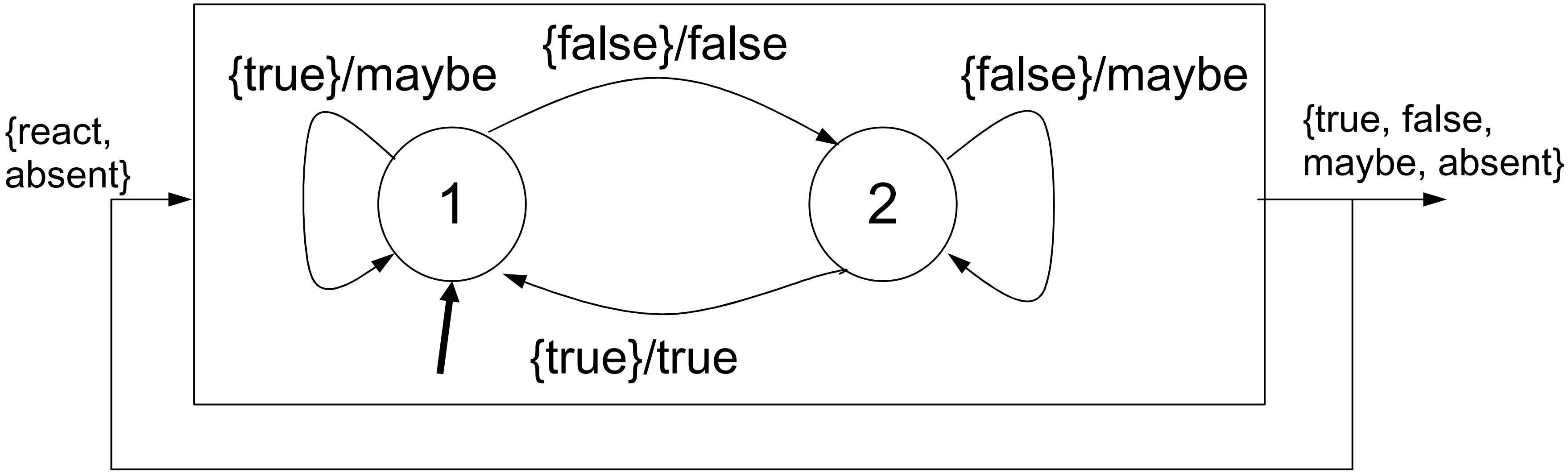


Consideration

- State-determined outputs is not a necessary condition for well-formedness
- The machine below is not state-determined ouput



Consideration

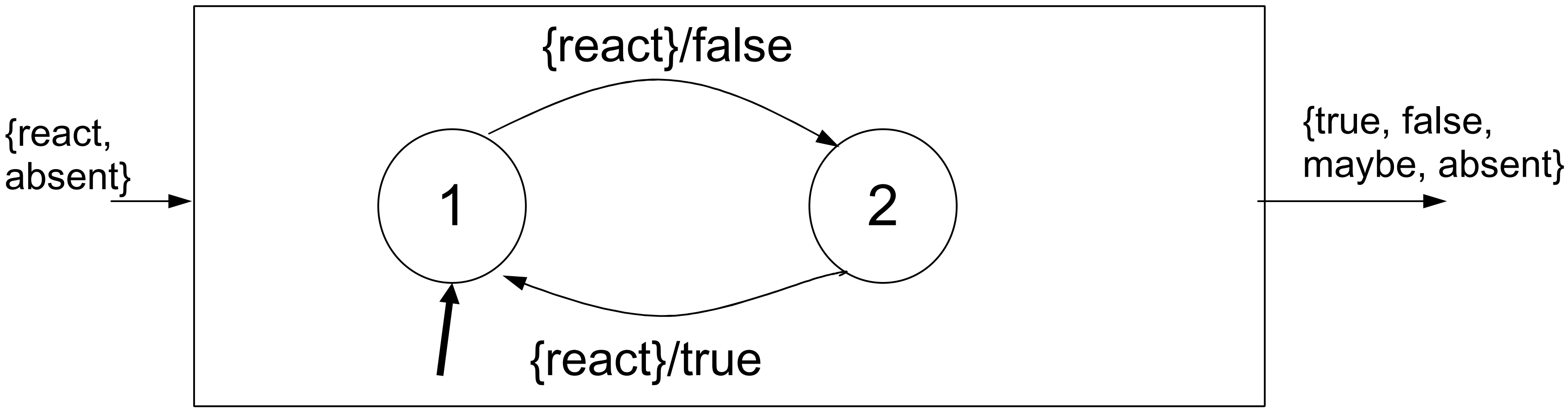


1 fp solution for each state

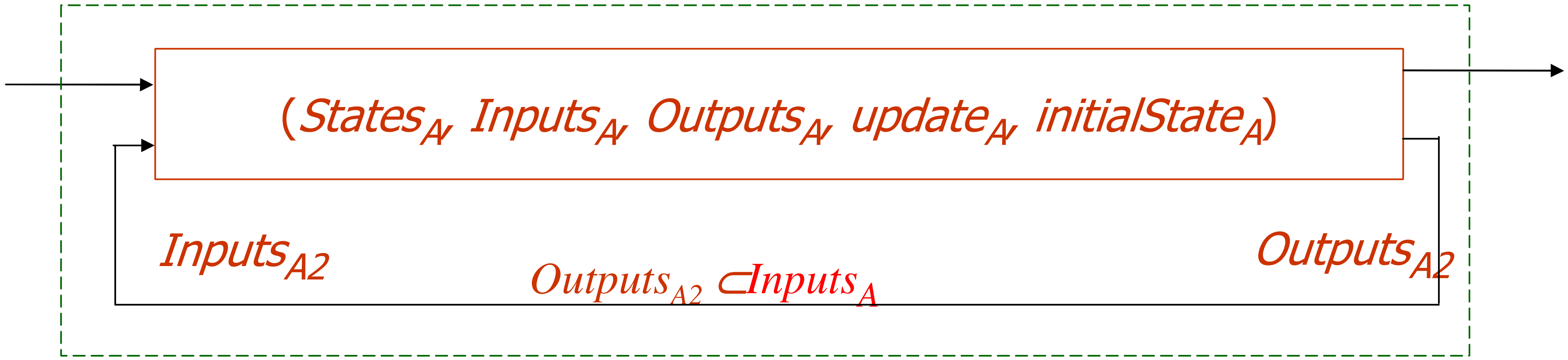
$$false = output_A(1, false)$$
$$true = output_A(2, true)$$

Consideration - (continued)

Equivalent machine



Feedback with inputs



- Inputs and outputs can be expressed in product form

$$Inputs_A = Inputs_{A1} \times Inputs_{A2}$$
$$Ouptus_A = Outputs_{A1} \times Outputs_{A2}$$
$$Outputs_{A2} \subset Inputs_{A_1}$$
$$output_A = (output_{A1}, output_{A2})\ where$$
$$ouput_{A1} : states_A \times Inputs_A \rightarrow Ouputs_{A1}$$
$$ouput_{A2} : states_A \times Iinputs_A \rightarrow Ouputs_{A2}$$

The FP problem

- The fixed problem is to find the unknown y=(y1,y2) s.t.

$$output_A(s(n),(x_1(n), y_2(n)))=(y_1(n), y_2(n))$$

which is equivalent to

$$output_{A1}(s(n),(x_1(n), y_2(n)))=y_1(n)$$
$$output_{A2}(s(n),(x_1(n), y_2(n)))=y_2(n)$$

Composition well formed if there is a unique solution to this!

x1 and s(n) are known: this is the crucial equation!

The FP problem - (continued)

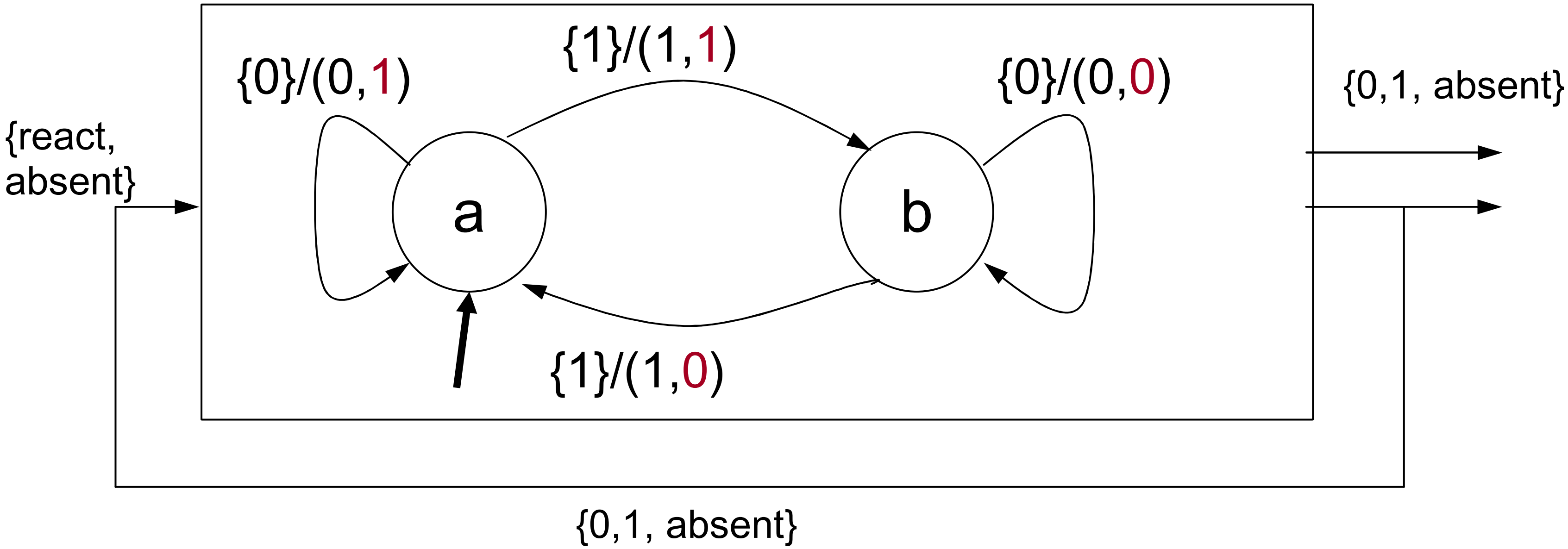
- If the composition is well formed it is described by

$$
\begin{gathered}
States = States_A \\
Inputs = Inputs_{A1} \\
Outputs = Outputs_{A1} \\
initialState = initialState_A \\
update(s(n), x(n)) = (nextState(s(n), x(n)), output(s(n), x(n))) \\
nextState(s(n), x(n)) = nextState_A(s(n), (x(n), y_2(n))) \\
output(s(n), x(n)) = output_A(s(n), (x(n), y_2(n))) \; where \\
y_2(n) = output_{A2}(s(n), (x(n), y_2(n)))
\end{gathered}
$$

Constructive Procedure

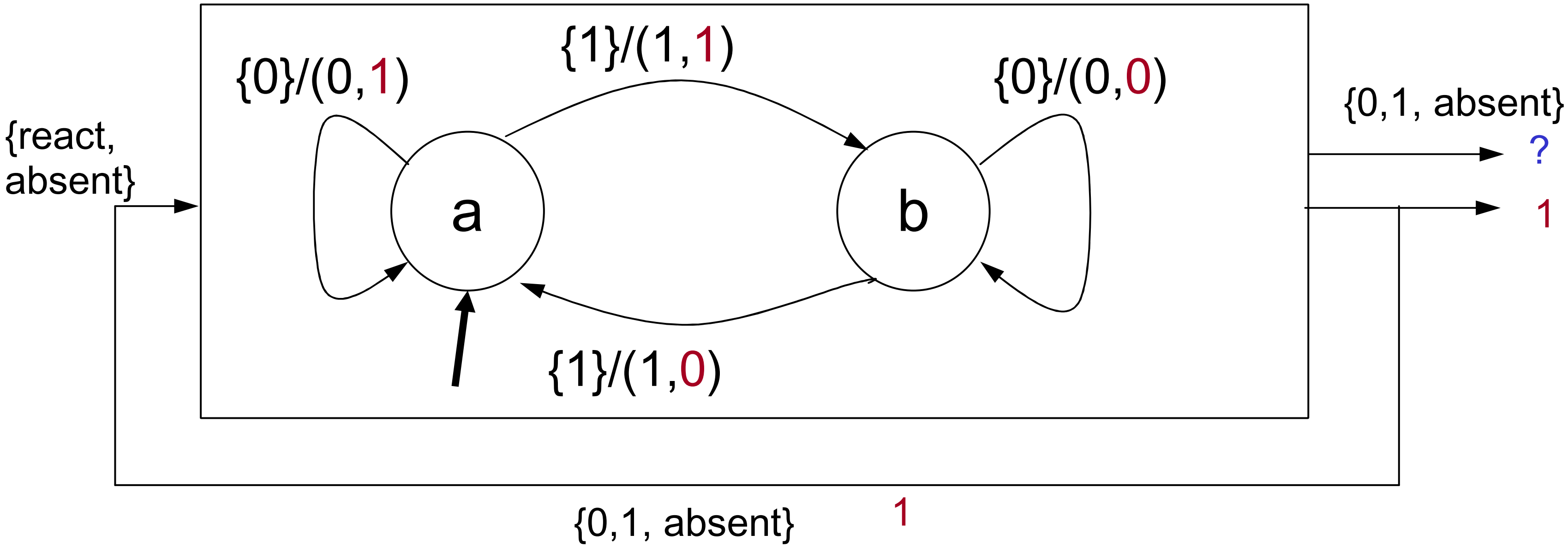
- The direct construction of a feedback composition is very difficult for many state machines
- Constructive procedure:
  - Begin with all unspecified signals *unknown*
  - Starting from any machine try to determine as much as possible on the outputs
  - Update the state machine with the newly acquired knowledge
  - Repeat the process until all signals are specified or there is nothing new to learn

Example



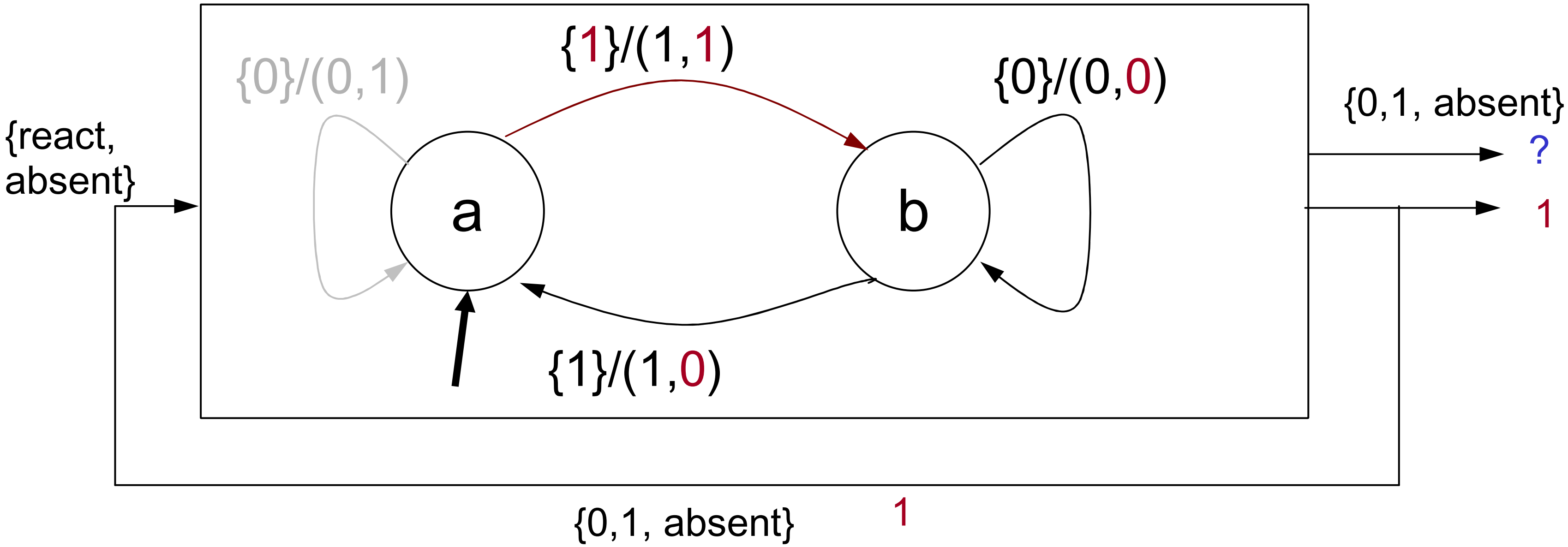
- We start assuming *unkwnown* symbol on the feedback loop
- Let's start from state *a*

Example - (continued)



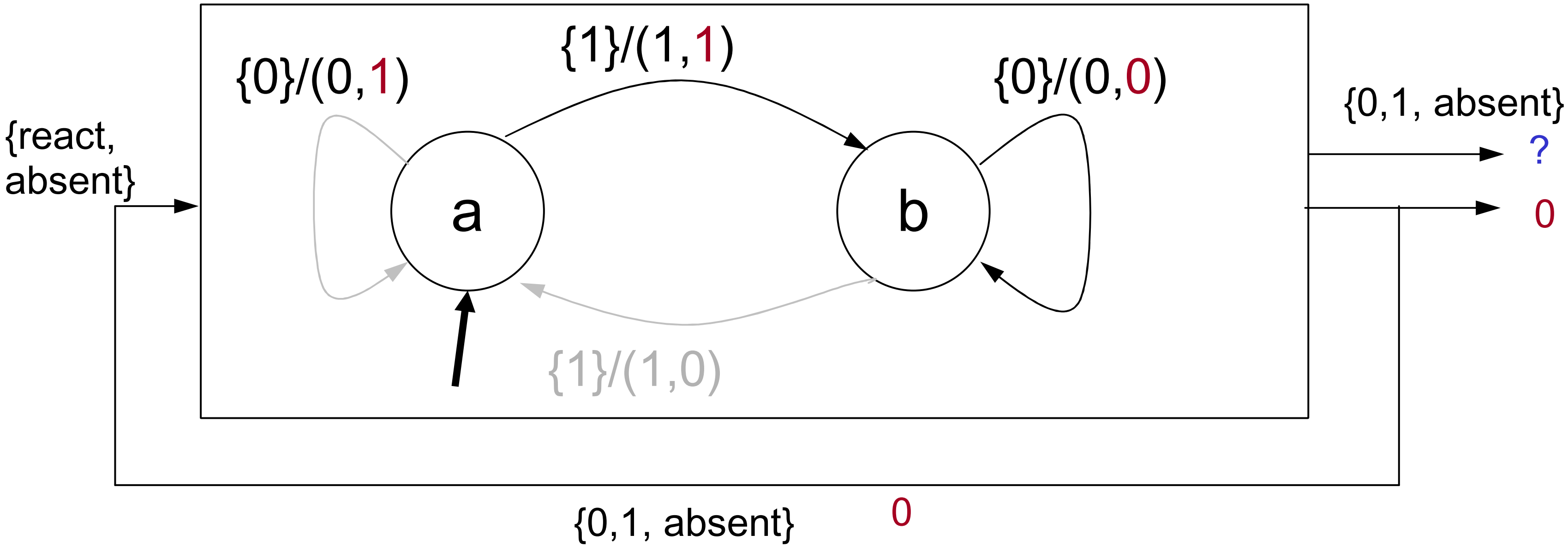
- The output is not state determined, however the second output is bound to be *1*
- The value of the feedback connection is changed from *unknown* to *1*

Example - (continued)



- Knowing the presence of *1* on the feedback loop a transition to state *b* is triggered
- We are done evaluating this transition and we move to state b

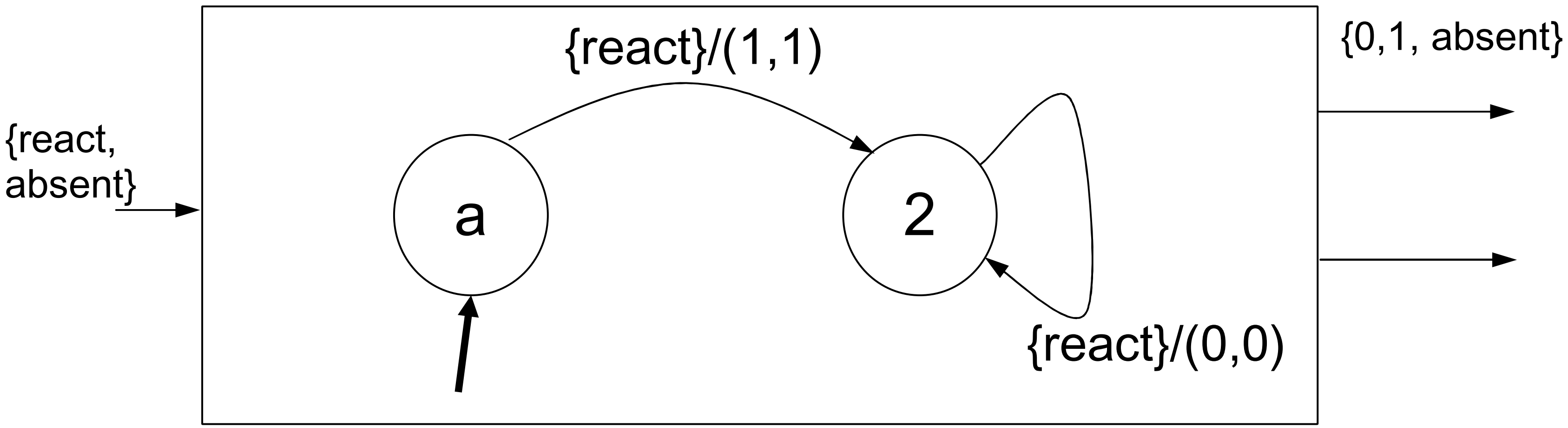
Example - (continued)



- In b the output is not state determined, however the second output is bound to be *0*
- The value of the feedback connection is changed from *unknown* to *0* and the only possible solution is the reaction back in b

Example

The equivalent machine



$$\{react,\ react,\ react,\ react,\ \ldots\ \} \rightarrow \{(1,1),\ (0,0),\ (0,0),\ (0,0),\ \ldots\}$$