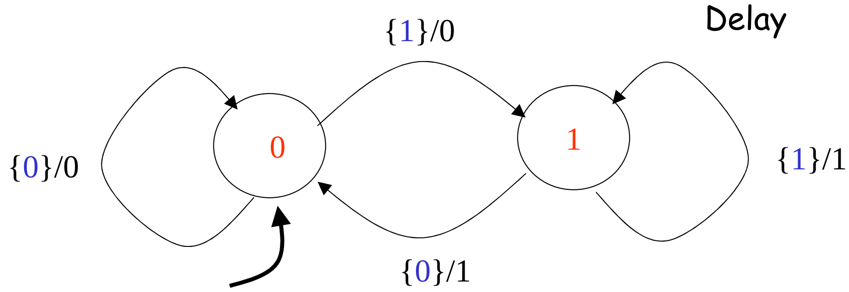
Week 4

1. More examples
2. Nondeterminism, equivalence, simulation (Ch 3)
3. Composition (Ch 4)

Delay



InputSignals = OutputSignals = [$Nats_0 \to$ {0,1, absent}]

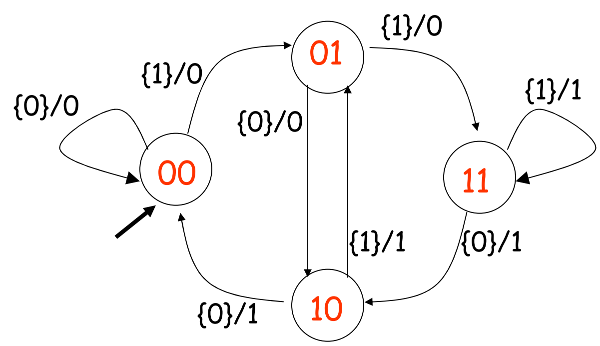
x = 0 0 1 1 0 …
s = 0 0 0 1 1 0 …
y = 0 0 0 1 1 0 …

$\forall x \in$ InputSignals, $\forall n \in Nats_0$, Delay(x)(n)=0, n=0;
= x(n-1), n > 0

$\forall x \in$ InputSignals, $\forall n \in Nats_0$,

$$Delay_2(x)(n) = 0, \quad n = 0,1;$$
$$= x(n-2), \quad n = 2,3,\ldots$$

Implement $Delay_2$ as state machine



We will see later that $Delay_2 \sim Delay_1 \circ Delay_1$

Nondeterministic state machines

In deterministic machines guards from state state are disjoint

In nondeterministic machines guards may not be disjoint. What does that mean?

Topics/determinism/example

The same input signal can lead to more than one state response and output signal

Set and function model

N = (States, Inputs, Outputs, possibleUpdates, initialState)

possibleUpdates: States x Inputs $\rightarrow$ P(States x Outputs)

where P(States x Outputs) is the set of all non-empty subsets of States x Outputs

Topics/deterministic/possible updates

Always: possibleUpdate(s, absent) = {(s,absent)}

A deterministic machine determines a function

H: InputSignals $\rightarrow$ OutputSignals

A nondeterministic machine determines a relation
Behaviors = {(x,y) | y is a possible output signal corresponding to x}
$\subset$ InputSignals x OutputSignals

Why non-deterministic machines?

1. Topics/determinism/Abstraction

2. Topics/determinism/Equivalence

3. Topics/determinism/Simulation

The matching game

Two (nondeterministic) machines,

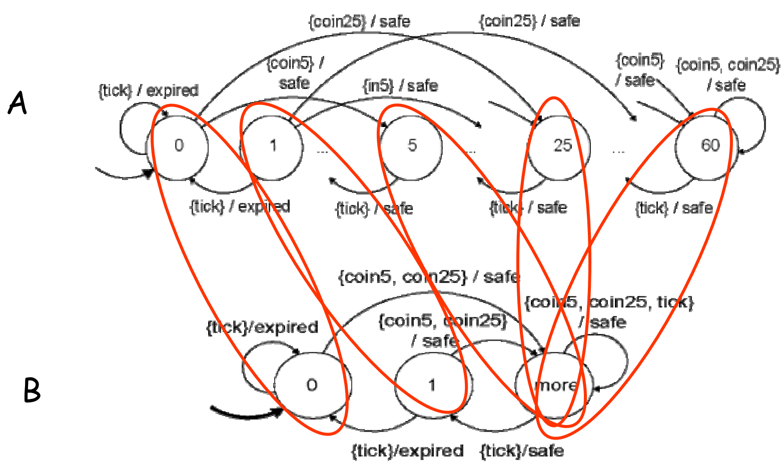
A = (States$_A$, Inputs, Outputs, possibleUpdates$_A$, $s_A(0)$)

B = (States$_B$, Inputs, Outputs, possibleUpdates$_B$, $s_B(0)$)

Suppose input symbol x and
A moves from $s_A(0)$ to $s_A(1)$ and produces output y

Then for same input symbol x
B can select move from $s_B(0)$ to $s_B(1)$, to produce y

and continue the game from states $s_A(1)$, $s_B(1)$



$S = \{(0_A, 0_B), (1_A, 1_B), (2_A, \text{more}), \ldots (60_A, \text{more})\}$

B simulates A if there is a subset

$S \subset \text{States}_A \times \text{States}_B$

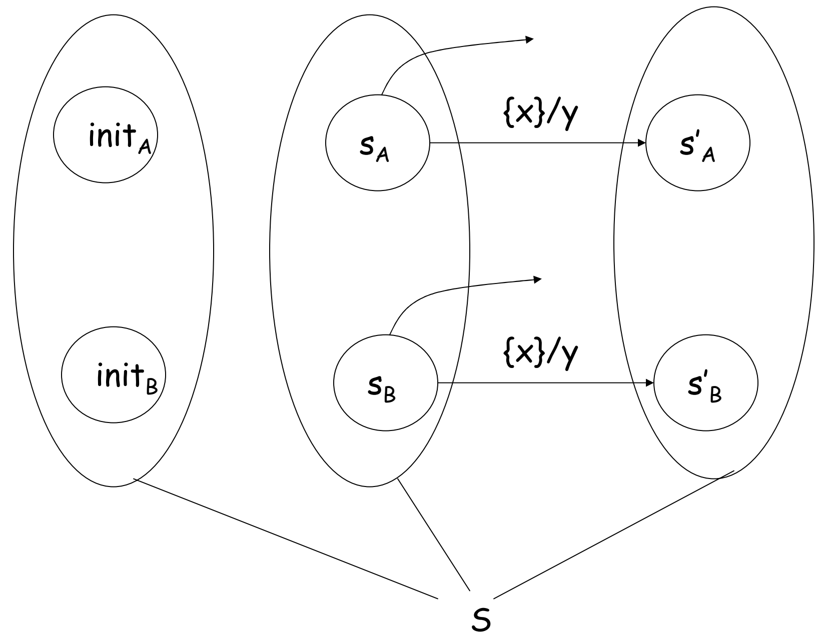
such that

1. $(\text{initialState}_A, \text{initialState}_B) \in S$, and

2. $\forall (s_A, s_B) \in S, \forall x \in \text{Inputs},$
   $\forall\ (s'_A, y) \in \text{possibleUpdates}_A(s_A, x)$

   $\exists\ (s'_B, y) \in \text{possibleUpdates}_B(s_B, x)$
   such that

   $(s'_A, s'_B) \in S$

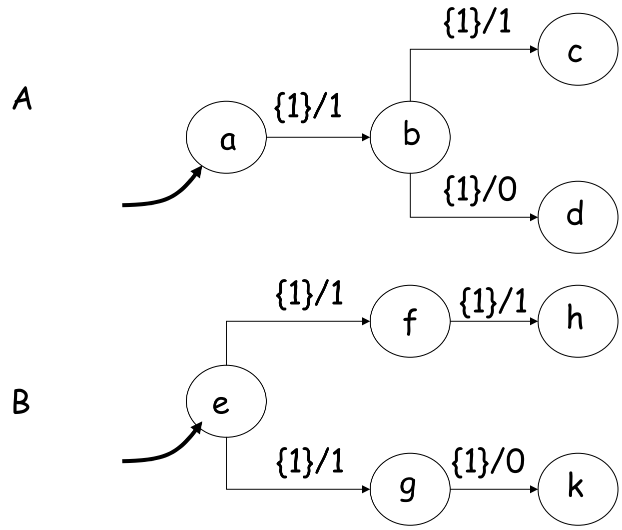


Theorem Suppose B simulates A. Then,

$$\text{Behaviors}_A \subset \text{Behaviors}_B$$

i.e. if y is a possible output response to x by machine A, y is also a possible output response to x by machine B.

Question Suppose B simulates A and C simulates B. Does C simulate A?



Behaviors of A = Behaviors of B. Why?

But B does not simulate A. Why?

Topics/Composition/Synchrony

1. Each component reacts once for every input symbol
2. The following happens simultaneously for each component
   - The input symbol is consumed
   - A state update occurs leading to next state and producing current output
   - If there is a feedback loop, the output appears at the input port

Topics/Composition/Side-by-side

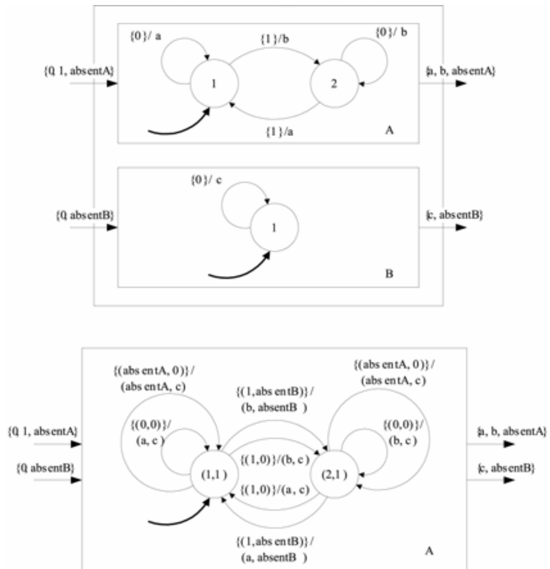
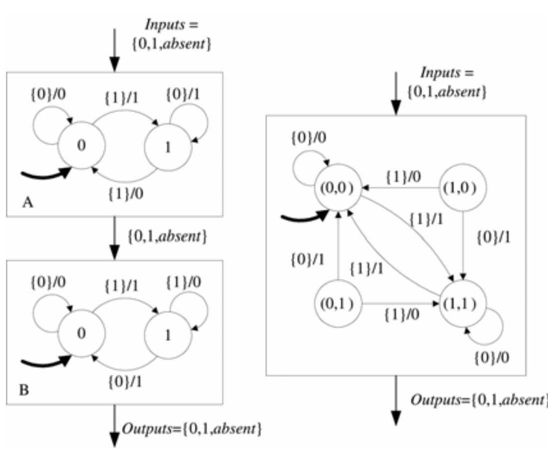


Fig 4.2, p. 127

Topics/Composition/Cascade



States (0,1) and (1,0) of cascade machine are NOT reachable

Fig 4.4, p 131

Topics/Composition/Productform

from audio storage

from ASIC interface to line

from tape machine

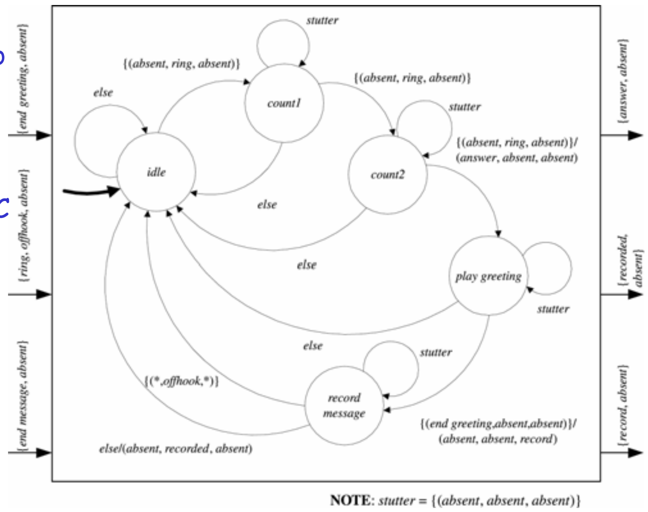


Fig 4.6, p. 134 Answering machine

Topics/Composition/Composition

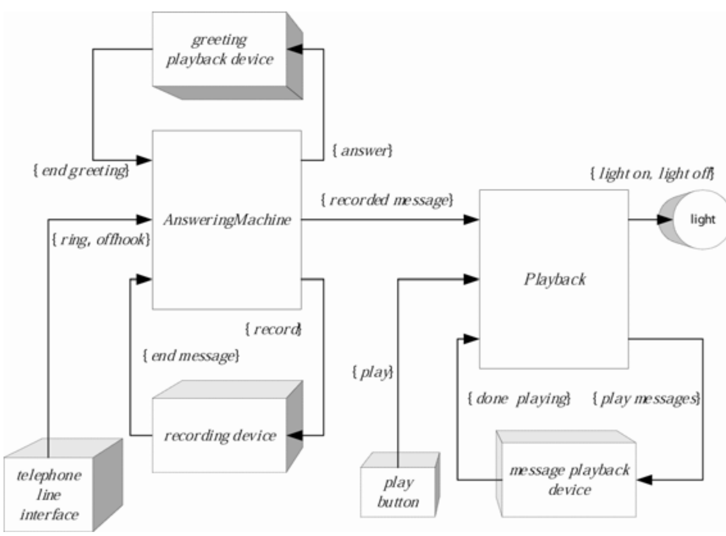


Fig 4.9, p. 136

Topics/Composition/Playback

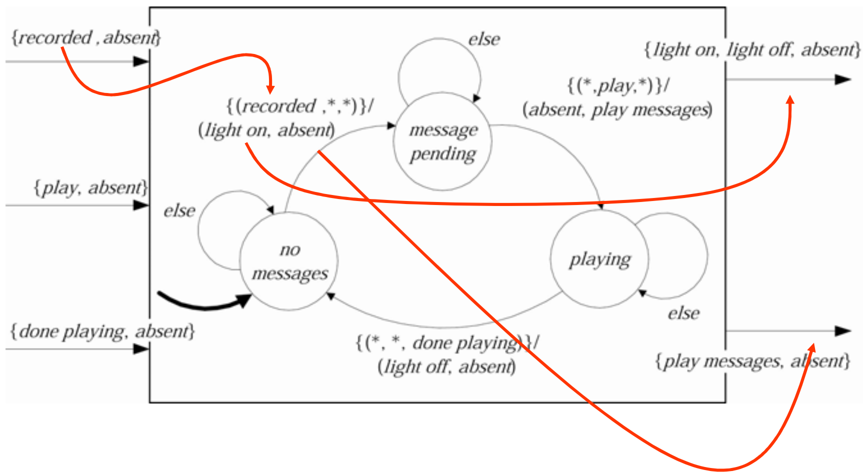
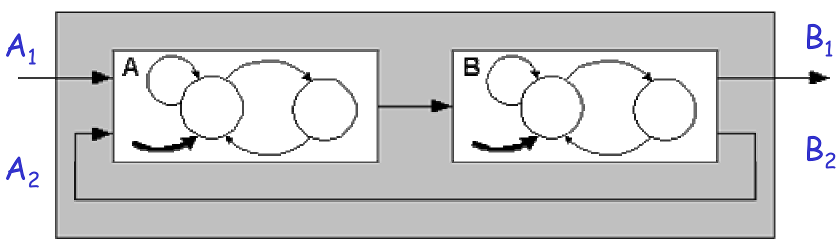


Fig 4.8, p. 136 Playback system

Feedback Composition



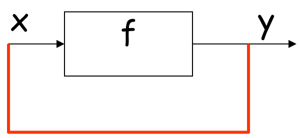
Basic assumptions:

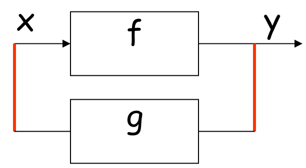
- $\text{Outputs}_A \subset \text{Inputs}_B$
- $\text{Outputs}_{B2} \subset \text{Inputs}_{A2}$

$\text{States} = \text{States}_A \times \text{States}_B$; $\text{Inputs} = \text{Inputs}_{A1}$; $\text{Outputs} = \text{Outputs}_{B1}$

Update function found by "fixed point" iteration

What is fixed point?





System: $y = f(x)$
Fdbk Connection: $x = y$
Must solve: $x = f(x)$

Systems: f,g
Fdbk Connection:
$x = g(y)$, $Y = f(x)$
Must solve: $x = g \circ f(x)$

No solution
More than one solution
Unique solution—well-formed

Fixed point example

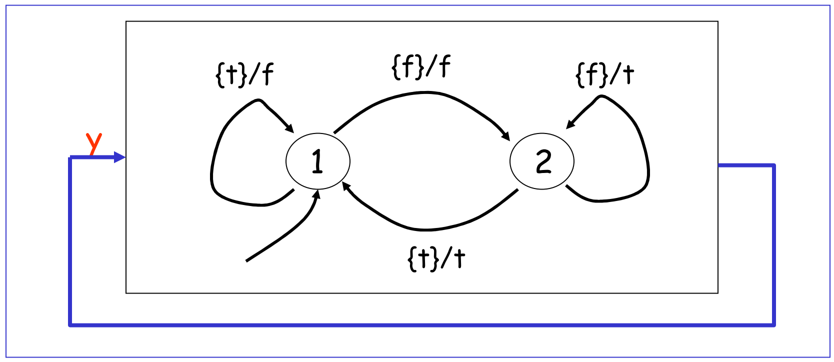
Take $X = Y =$ Reals, and f given by
for all x, $f(x) = 2x - 1$

So fixed point equation is:

$$2x - 1 = x$$

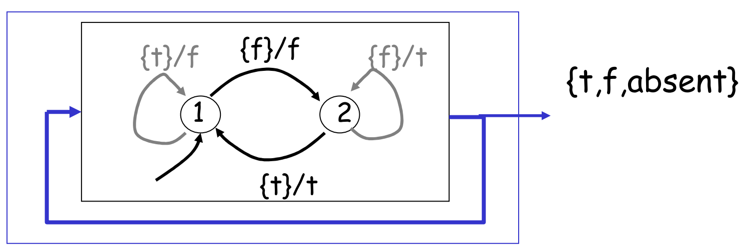
which has unique solution $x = 1$

Feedback w/o external input

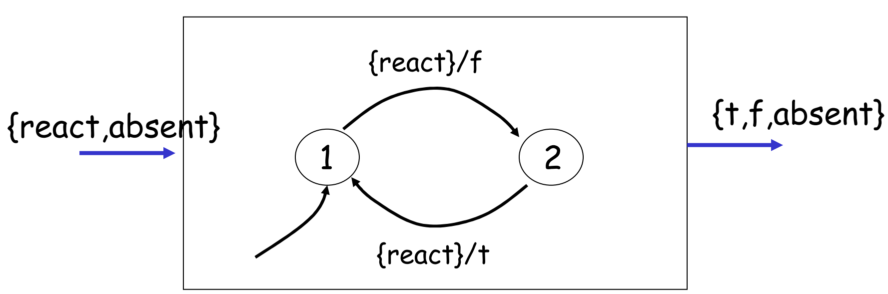


output(1, y) = y   y = f, unique

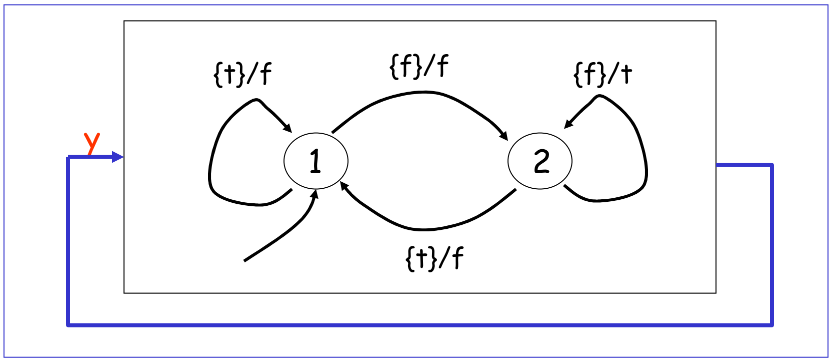
output(2, y) = y   y = t, unique



is equivalent to



Feedback w/o input

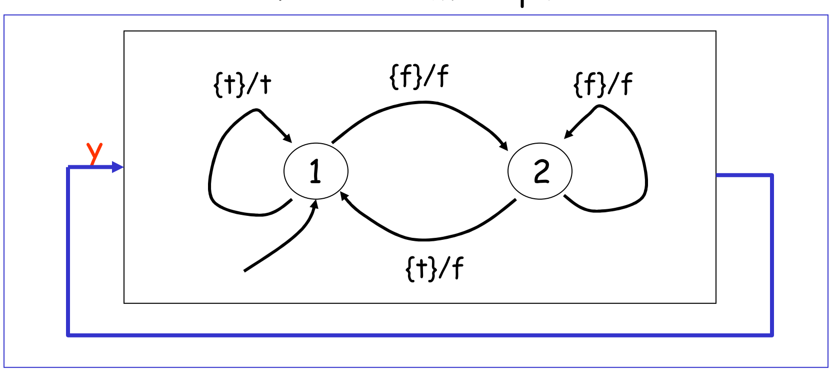


Update(1, y) = (s, y)   y = f, s = 2, unique

Update(2, y) = (s, y)   No solution

Feedback connection not well-formed

Feedback w/o input

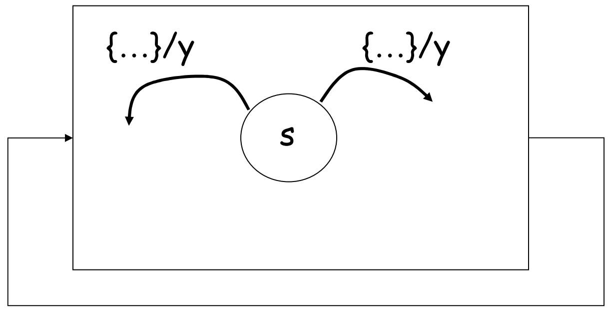


output(1, y) = y   y=f, y=t, non-unique

output(2, y) = y   y=f, unique

Feedback connection not well-formed

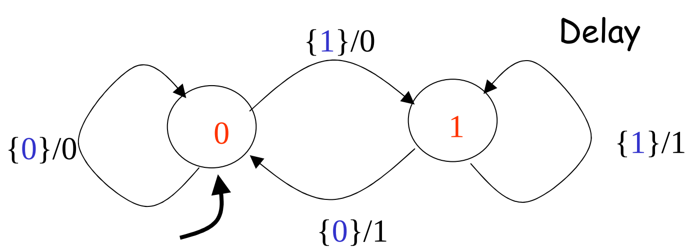
State-determined output gives well-formed fdbk connection

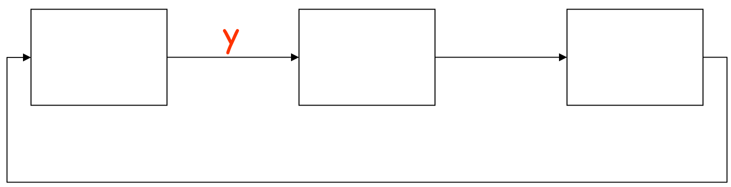


If output in current state is independent of input,

output(s, y) = y has unique solution

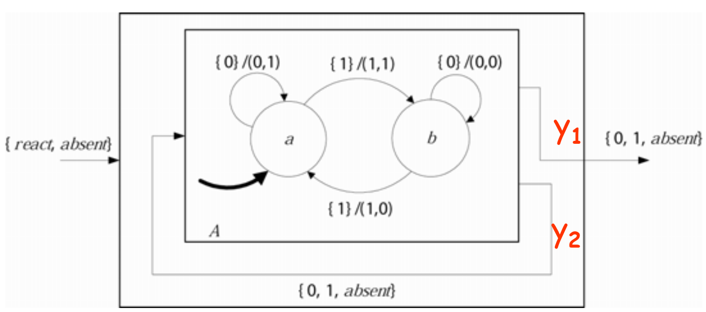
Delay machine has state-determined output



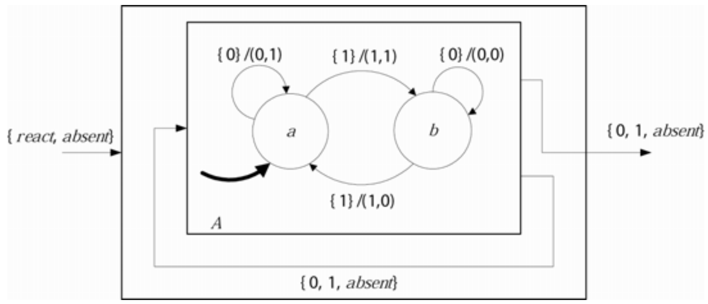


In general start with unknown output y anywhere

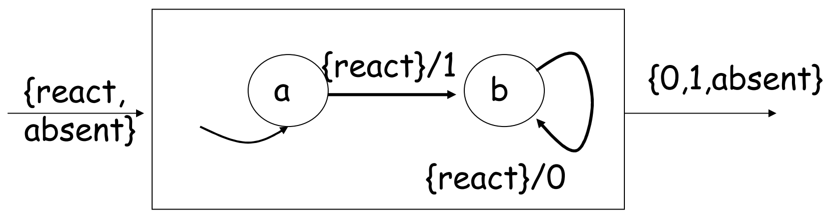
- For each machine
  - If output can be determined, produce it
  - If state transition can be determined, take it
- Repeat until no progress can be made
- If all outputs are determined – well-formed
- If some signals are unknown– not well-formed



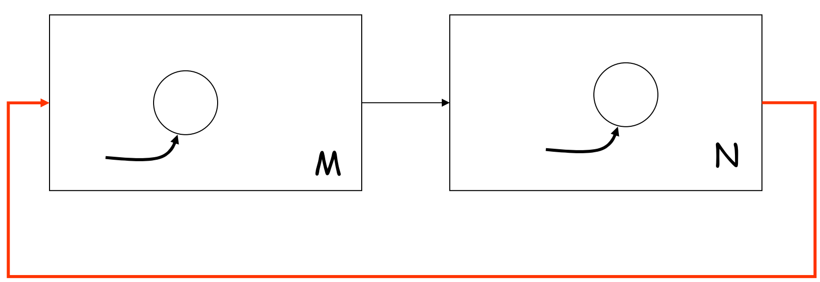
1. Start with state a and unknown y = (y_1, y_2)
2. Y is not determined, but y_2 = 1
3. Start with state a and y = (y_1, 1), then must have
   update (a, 1) = (b, (1,1)).
4. Start with state b and unknown y = (y_1, y_2)
5. Y is not determined, but y_2 = 0
6. Start with state b and y = (y_1, 0), then must have
   update (b, 0) = (b, (0,0))



is equivalent to

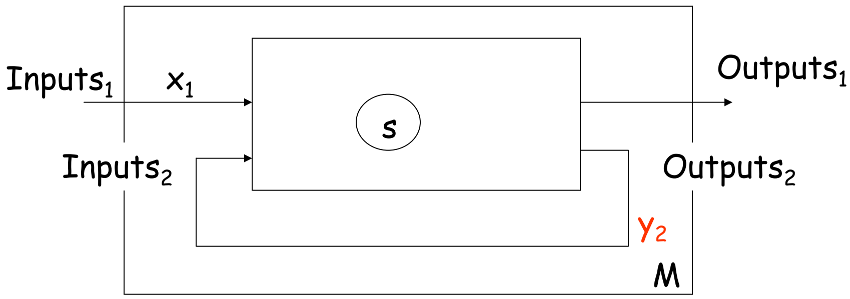


State-determined output



If M has state-determined output, the feedback connection is well-formed, no matter what N is

Feedback with input



$Outputs_2 \subset Inputs_1$

Start with state s

$\forall x_1 \in Inputs_1$ solve: $output_{M2}(s, (x_1, y_2)) = y_2$

If y_2 is unique, take

$nextState(s, x_1) = (nextState_M(s, (x_1, y_2))$

$output(s, x_1) = output_{M1}(x_1, y_2)$