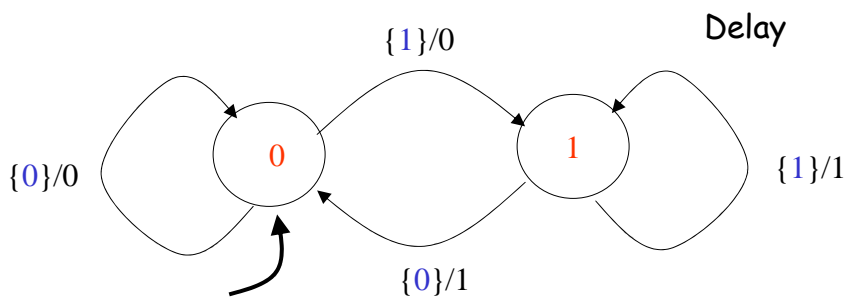


Week 4

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



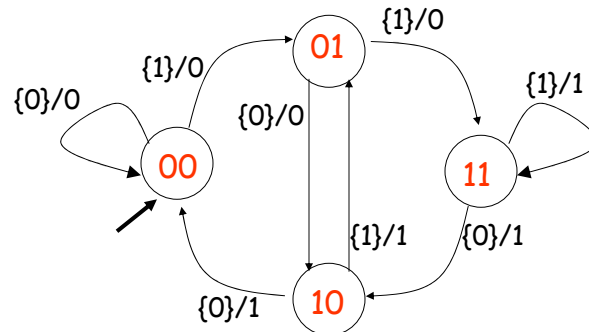
$\text{InputSignals} = \text{OutputSignals} = [\text{Nats}_0 \rightarrow \{0,1, \text{absent}\}]$

$x = 00110 \dots$
 $s = 000110 \dots$
 $y = 000110 \dots$

$\forall x \in \text{InputSignals}, \forall n \in \text{Nats}_0, \text{Delay}(x)(n) = 0, \quad n=0;$
 $\quad \quad \quad = x(n-1), \quad n > 0$

$\forall x \in \text{InputSignals}, \forall n \in \text{Nats}_0,$
 $\text{Delay}_2(x)(n) = 0, \quad n = 0,1;$
 $\quad = x(n-2), \quad n = 2,3,\dots$

Implement Delay_2 as state machine



We will see later that $\text{Delay}_2 \sim \text{Delay}_1 \circ \text{Delay}_1$

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

$N = (\text{States}, \text{Inputs}, \text{Outputs}, \text{possibleUpdates}, \text{initialState})$

$\text{possibleUpdates}: \text{States} \times \text{Inputs} \rightarrow P(\text{States} \times \text{Outputs})$

where $P(\text{States} \times \text{Outputs})$ is the set of all non-empty subsets of $\text{States} \times \text{Outputs}$

Topics/deterministic/possible updates

Always: $\text{possibleUpdate}(s, \text{absent}) = \{(s, \text{absent})\}$

A deterministic machine determines a function

$H: \text{InputSignals} \rightarrow \text{OutputSignals}$

A nondeterministic machine determines a relation

$\text{Behaviors} = \{(x, y) \mid y \text{ is a possible output signal corresponding to } x\}$

$\subset \text{InputSignals} \times \text{OutputSignals}$

Why non-deterministic machines?

1. Topics/determinism/Abstraction

2. Topics/determinism/Equivalence

3. Topics/determinism/Simulation

The matching game

Two (nondeterministic) machines,

$A = (\text{States}_A, \text{Inputs}, \text{Outputs}, \text{possibleUpdates}_A, s_A(0))$

$B = (\text{States}_B, \text{Inputs}, \text{Outputs}, \text{possibleUpdates}_B, s_B(0))$

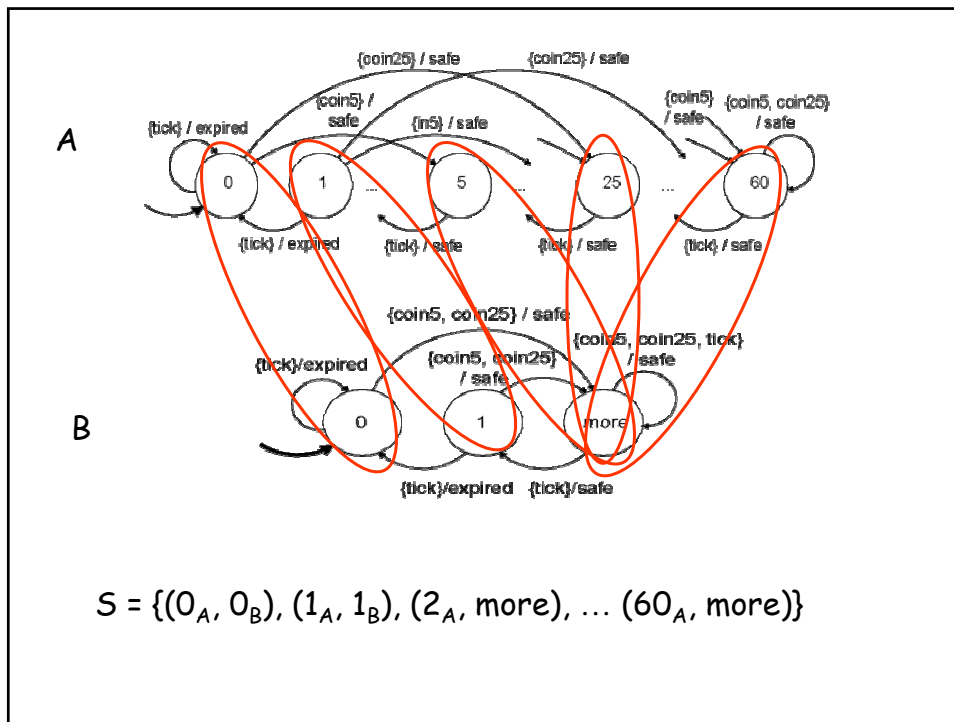
Suppose input symbol x and

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 $s_A(1), s_B(1)$



B **simulates** A if there is a subset

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

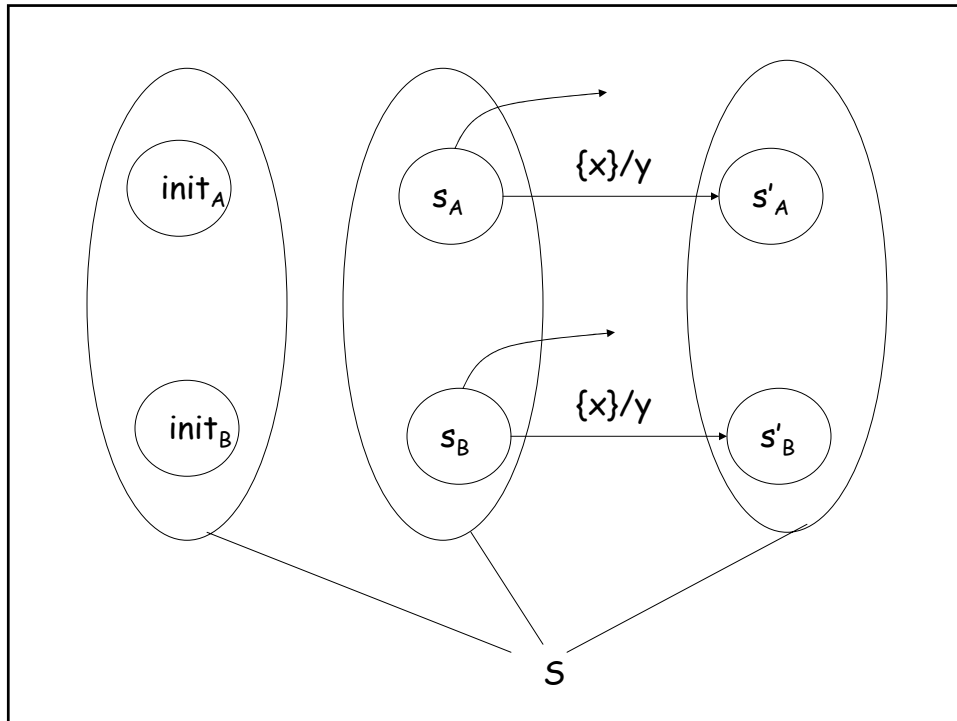
such that

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

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

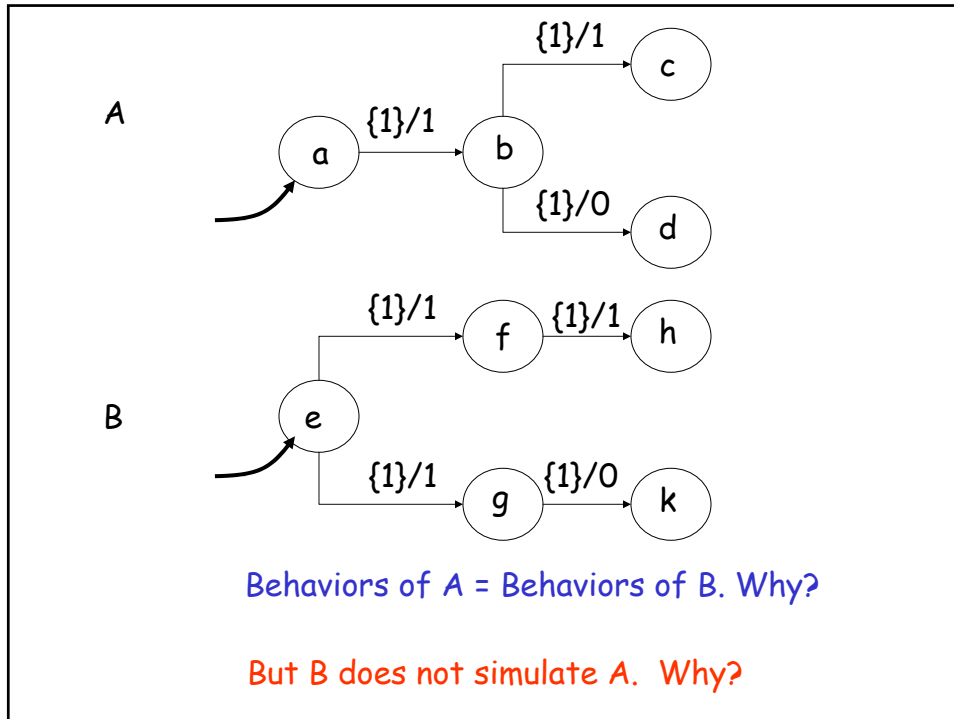


Theorem Suppose B simulates A. Then,

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

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?



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

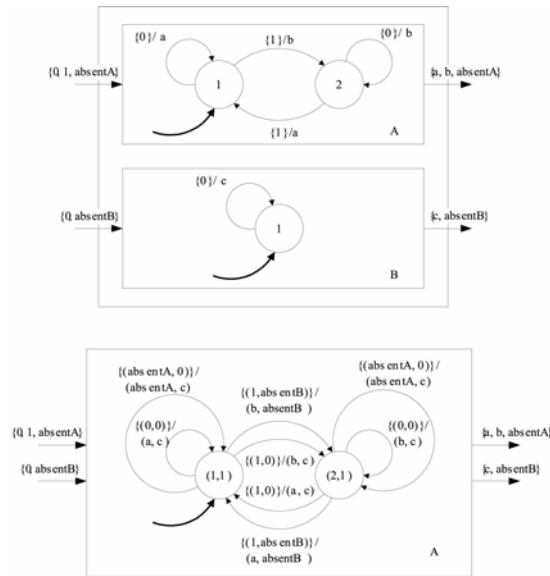
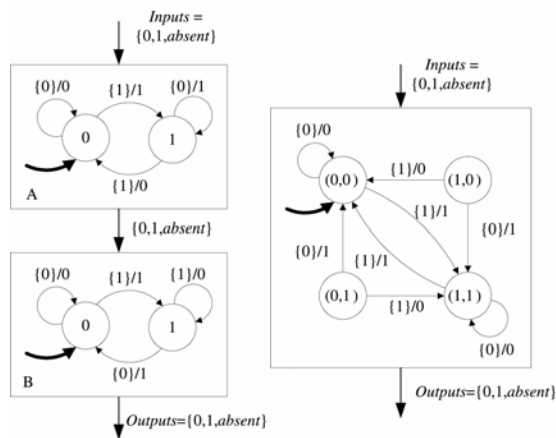


Fig 4.2, p. 127

Topics/Composition/Cascade



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

Fig 4.4, p 131

Topics/Composition/Productform

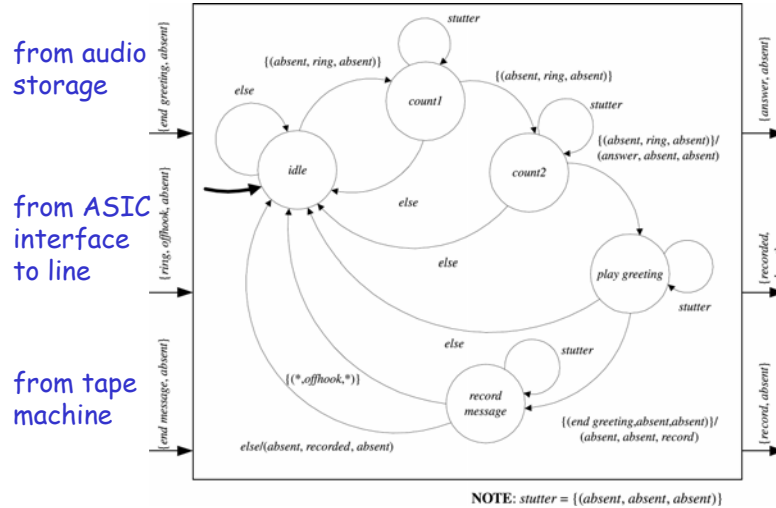


Fig 4.6, p. 134 Answering machine

Topics/Composition/Composition

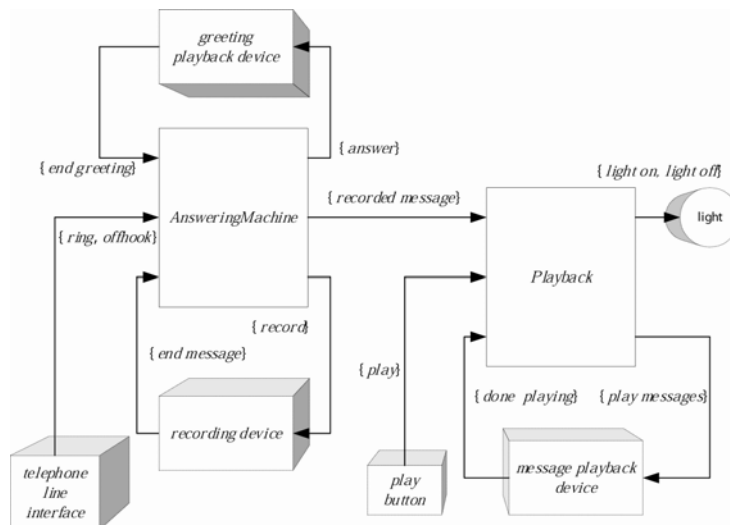


Fig 4.9, p. 136

Topics/Composition/Playback

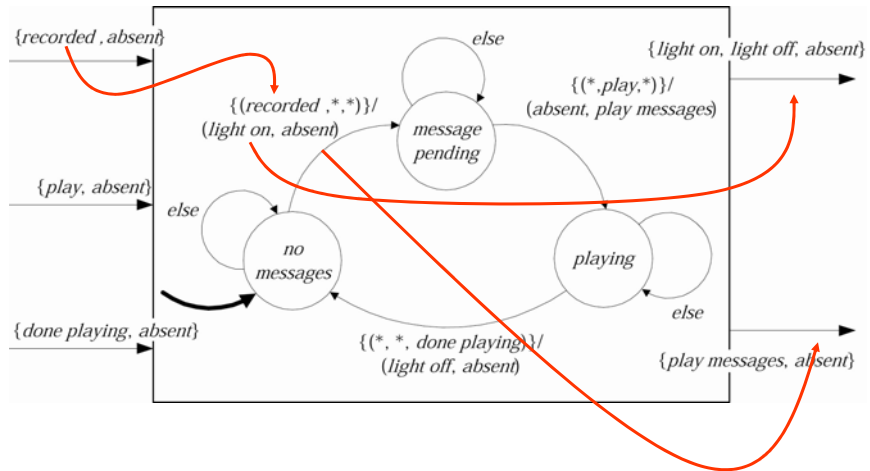
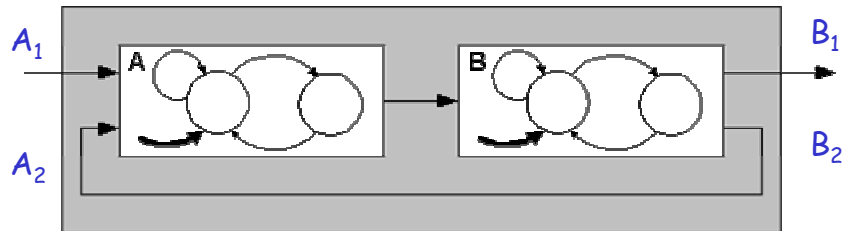


Fig 4.8, p. 136 Playback system

Feedback Composition



Basic assumptions:

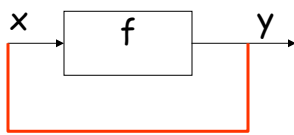
- $\text{Outputs}_A \subset \text{Inputs}_B$
- $\text{Outputs}_B \subset \text{Inputs}_A$

States = $\text{States}_A \times \text{States}_B$; Inputs = Inputs_{A1} ;

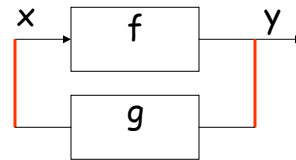
Outputs = Outputs_{B1}

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

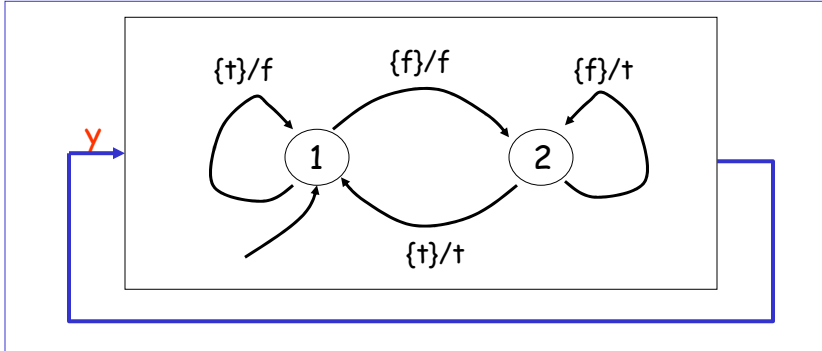
Take $X = Y = \text{Reals}$, and f given by
for all x , $f(x) = 2x - 1$

So fixed point equation is:

$$2x - 1 = x$$

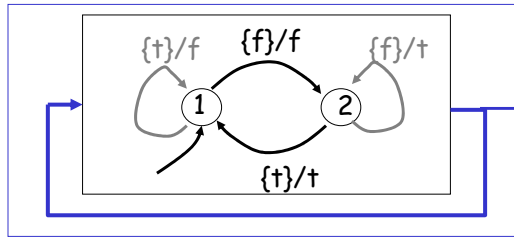
which has unique solution $x = 1$

Feedback w/o external input



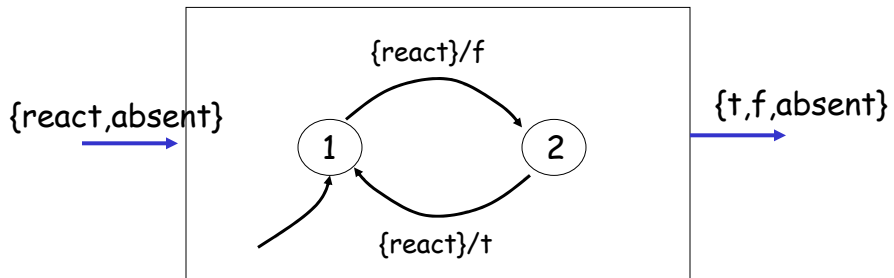
output(1, y) = y $y = f$, unique

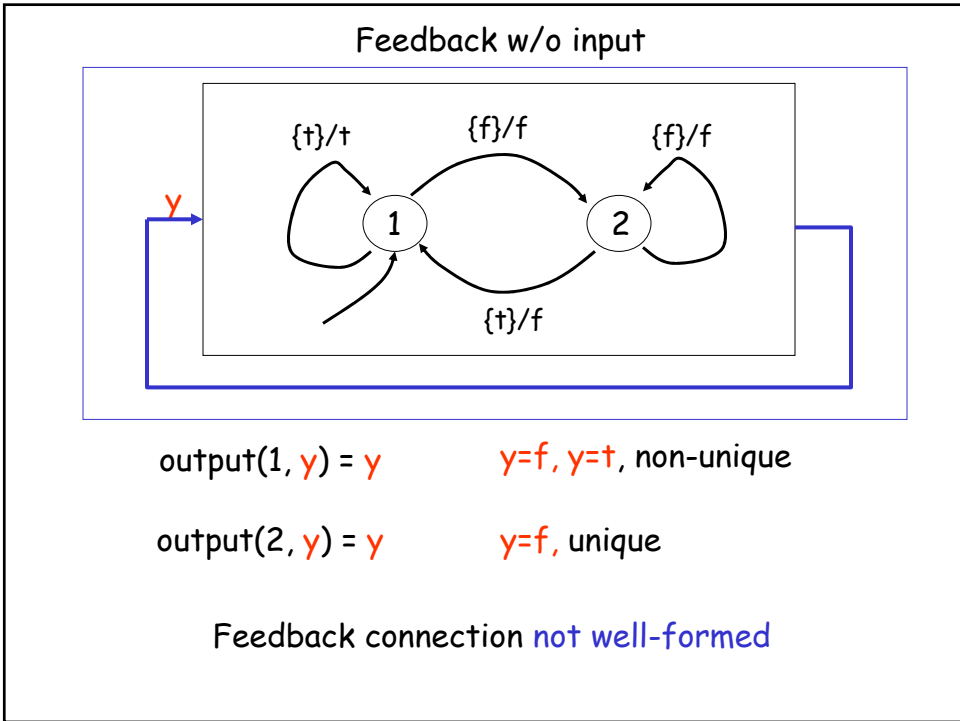
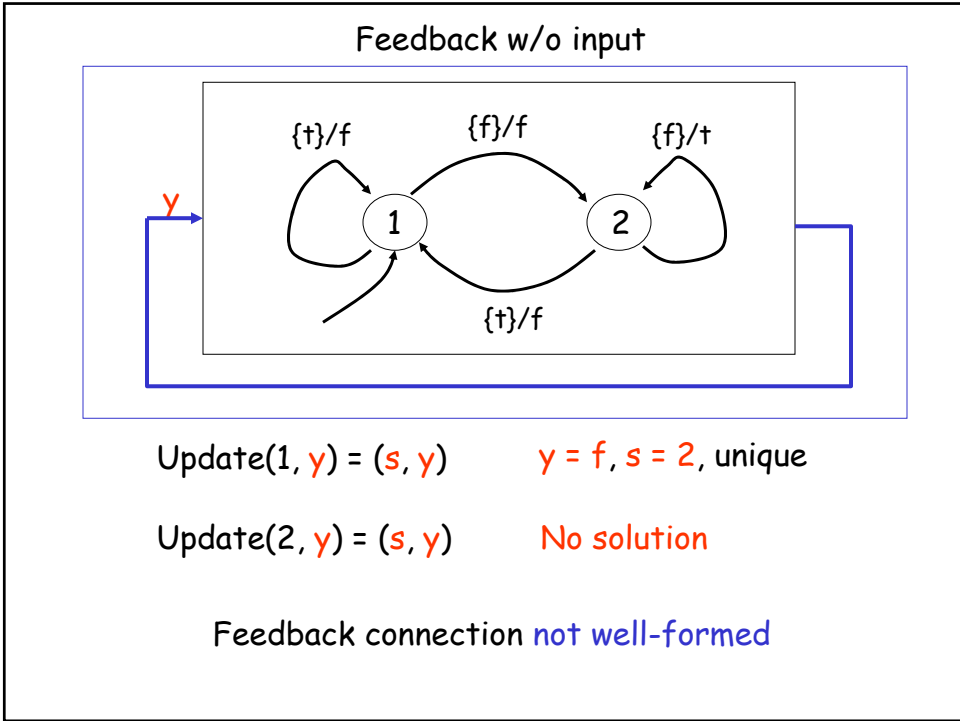
output(2, y) = y $y = t$, unique



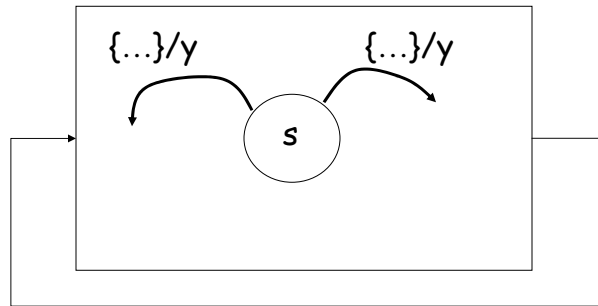
{t,f,absent}

is equivalent to



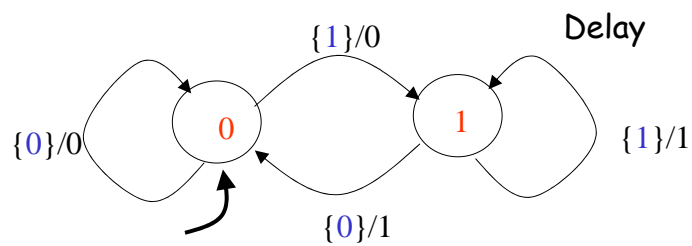


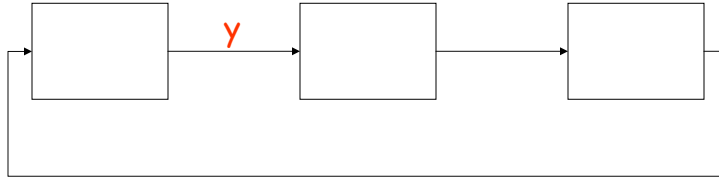
State-determined output gives well-formed fdbk connection



If output in current state is independent of input,
 $\text{output}(s, y) = y$ has unique solution

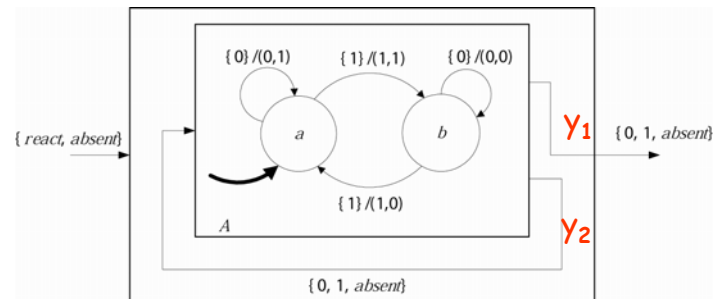
Delay machine has state-determined output





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 $(b, 0) = (b, (0,0))$**

