Product Machines

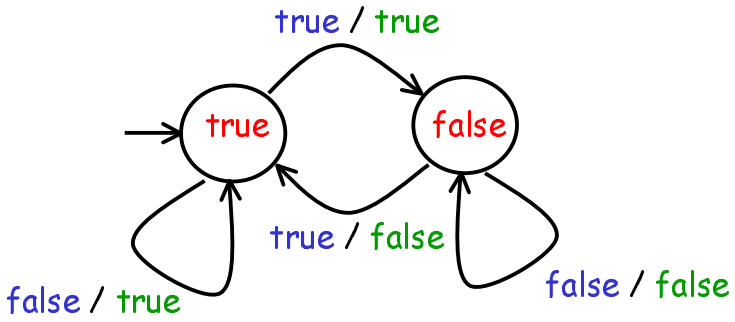
EECS 20

Lecture 10 (February 7, 2001)

Tom Henzinger

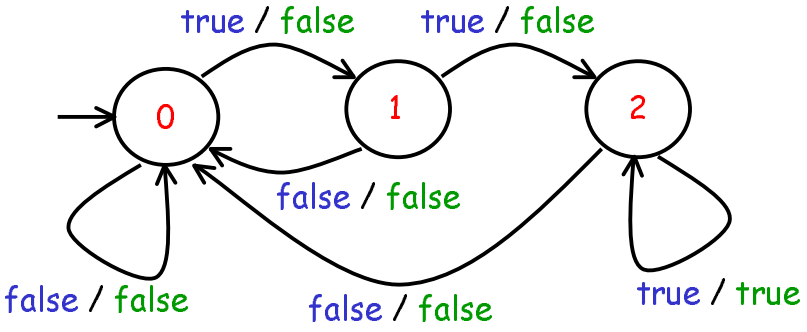
Composition of State Machines

Transition Diagram of the Parity System



States = Bools
Inputs = Bools
Outputs = Bools

Transition Diagram of the LastThree System



States = { 0, 1, 2 }
Inputs = Bools
Outputs = Bools

The Parity System :

States [Parity] = { true, false }

initialState [Parity] = true

nextState [Parity] (q,x) = $(q \neq x)$

output [Parity] (q,x) = q

The LastThree System :

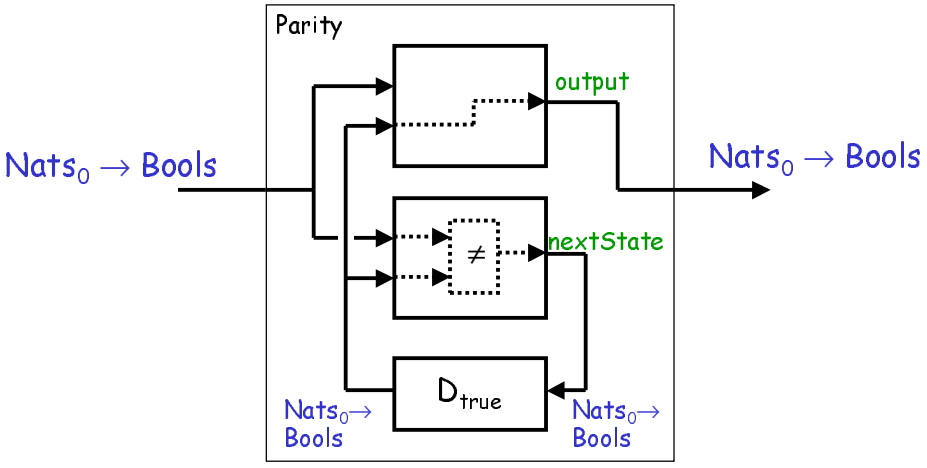
States [LastThree] = { 0, 1, 2 }

initialState [LastThree] = 0

nextState [LastThree] (q,x) = $\begin{cases} 0 & \text{if } \neg x \\ \min(q+1, 2) & \text{if } x \end{cases}$

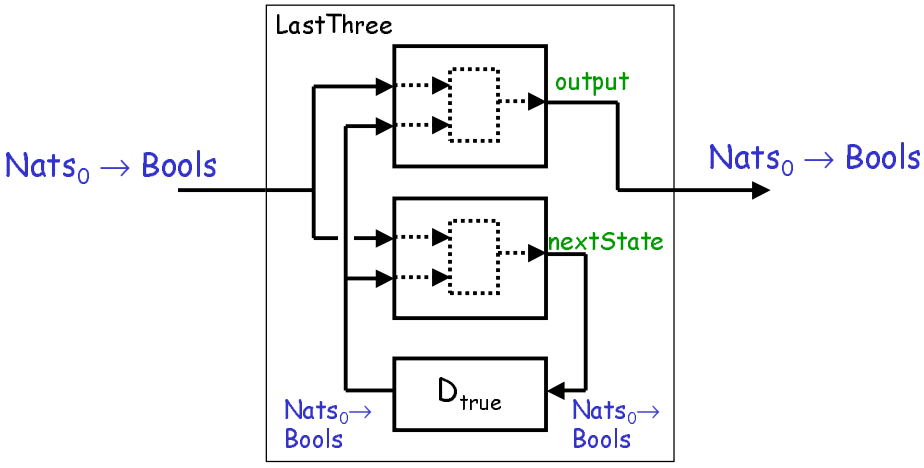
output [LastThree] (q,x) = $((q = 2) \wedge x)$

Block Diagram of Parity System

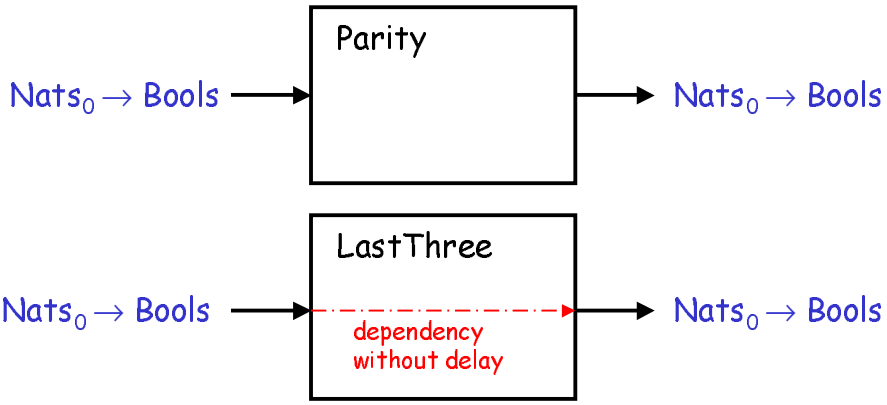


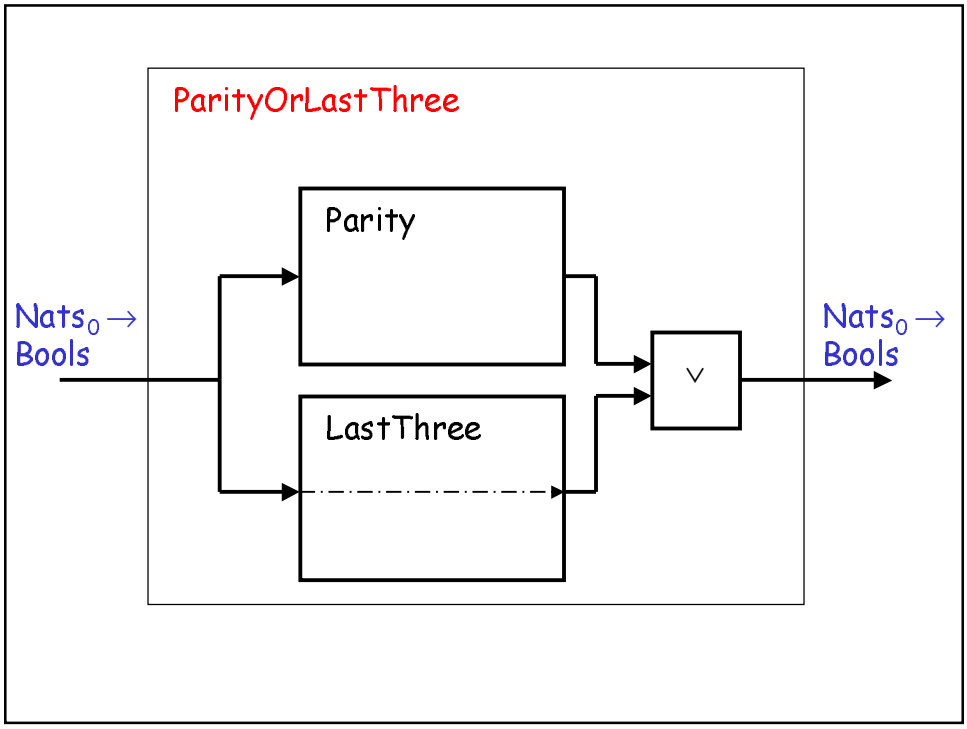
No path without delay from input to output.

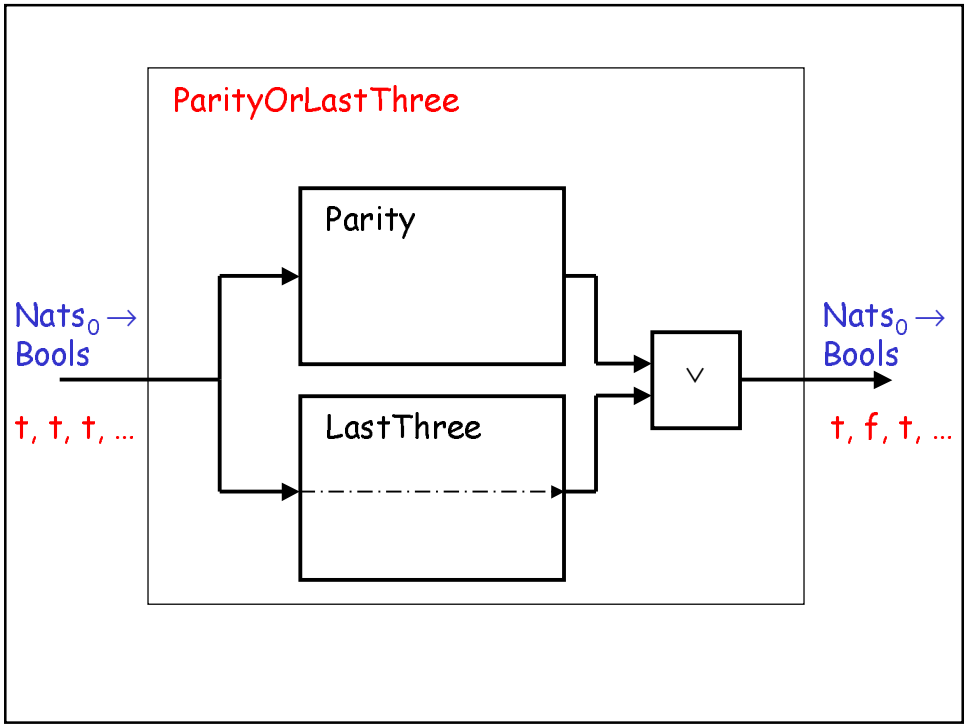
Block Diagram of LastThree System



Path without delay from input to output !




ParityOrLastThree
Parity
LastThree
∨
Nats$_0$ →
Bools
Nats$_0$ →
Bools


ParityOrLastThree
Parity
LastThree
∨
Nats$_0$ →
Bools
t, t, t, …
Nats$_0$ →
Bools
t, f, t, …

The ParityOrLastThree System

Inputs [ParityOrLastThree] = Bools

Outputs [ParityOrLastThree] = Bools

States [ParityOrLastThree]

= States [Parity] × States [LastThree]

= { true, false } × { 0, 1, 2 }

initialState [ParityOrLastThree]

= (initialState [Parity], initialState [LastThree])

= (true, 0)

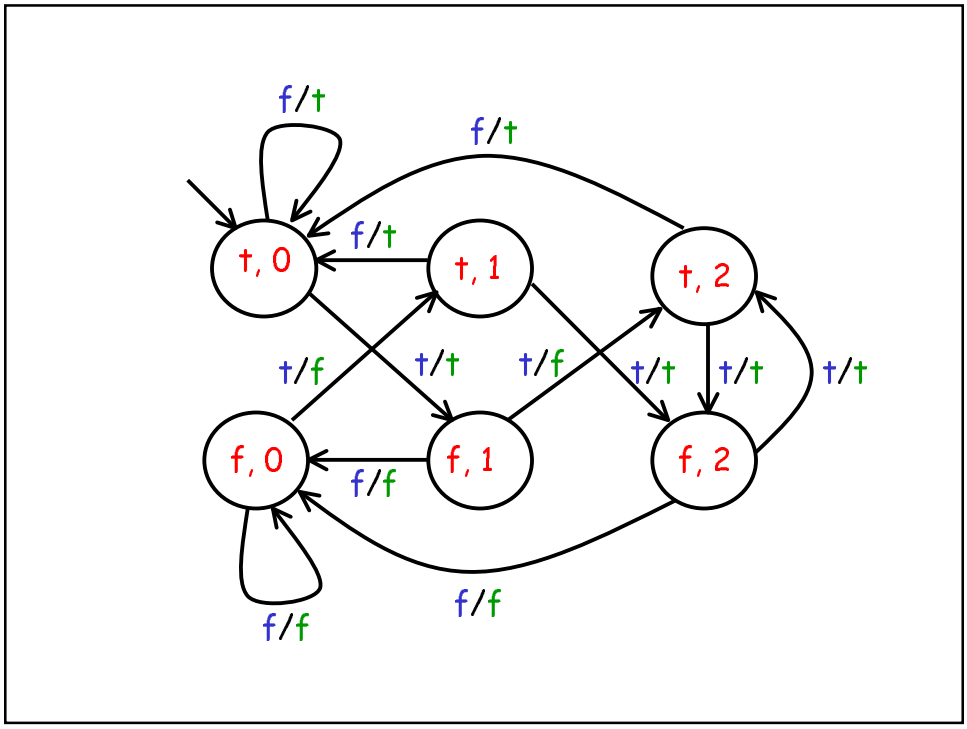
The ParityOrLastThree System, continued

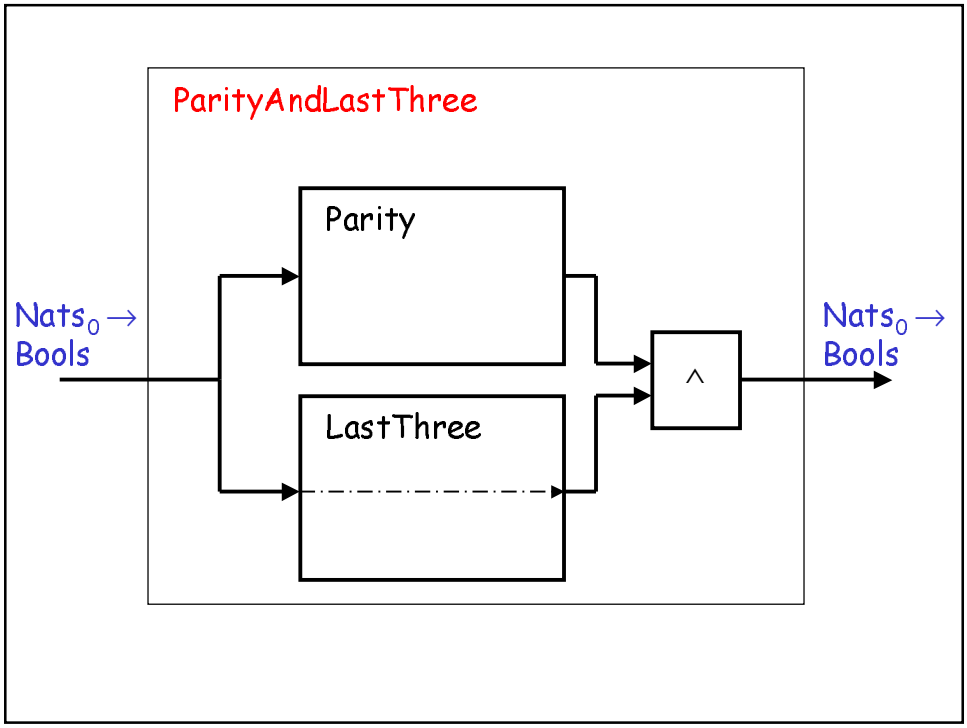
nextState [ParityOrLastThree] ((q1, q2) , x)

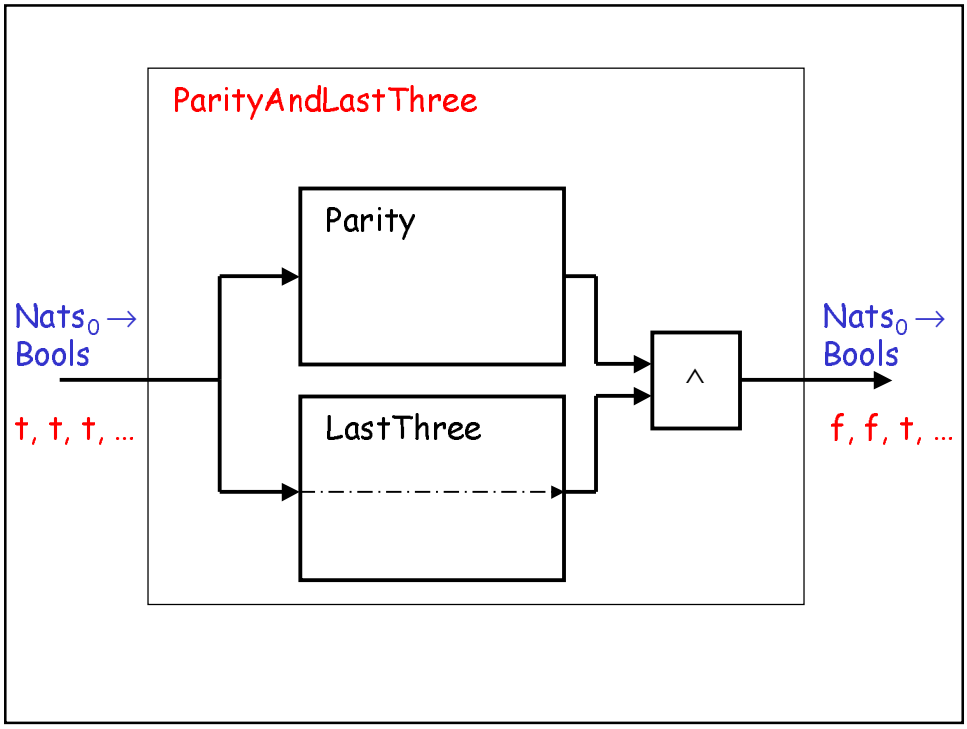
= (nextState [Parity] (q1, x) , nextState [LastThree] (q2, x))

output [ParityOrLastThree] ((q1, q2) , x)

= output [Parity] (q1, x) ∨ output [LastThree] (q2, x)


f/t
f/t
f/t
t, 0
t, 1
t, 2
t/f
t/t
t/f
t/t
t/t
t/t
f, 0
f, 1
f, 2
f/f
f/f
f/f


ParityAndLastThree
Parity
LastThree
^
Nats0 → Bools
Nats0 → Bools



The ParityAndLastThree System

Inputs [ParityAndLastThree] = Bools

Outputs [ParityAndLastThree] = Bools

States [ParityAndLastThree]

= States [Parity] × States [LastThree]

= { true, false } × { 0, 1, 2 }

initialState [ParityAndLastThree]

= (initialState [Parity], initialState [LastThree])

= (true, 0)

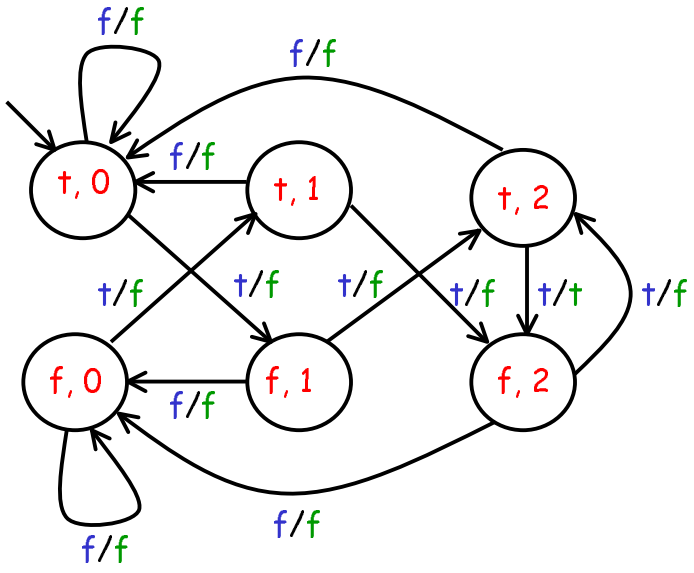
The ParityAndLastThree System, continued

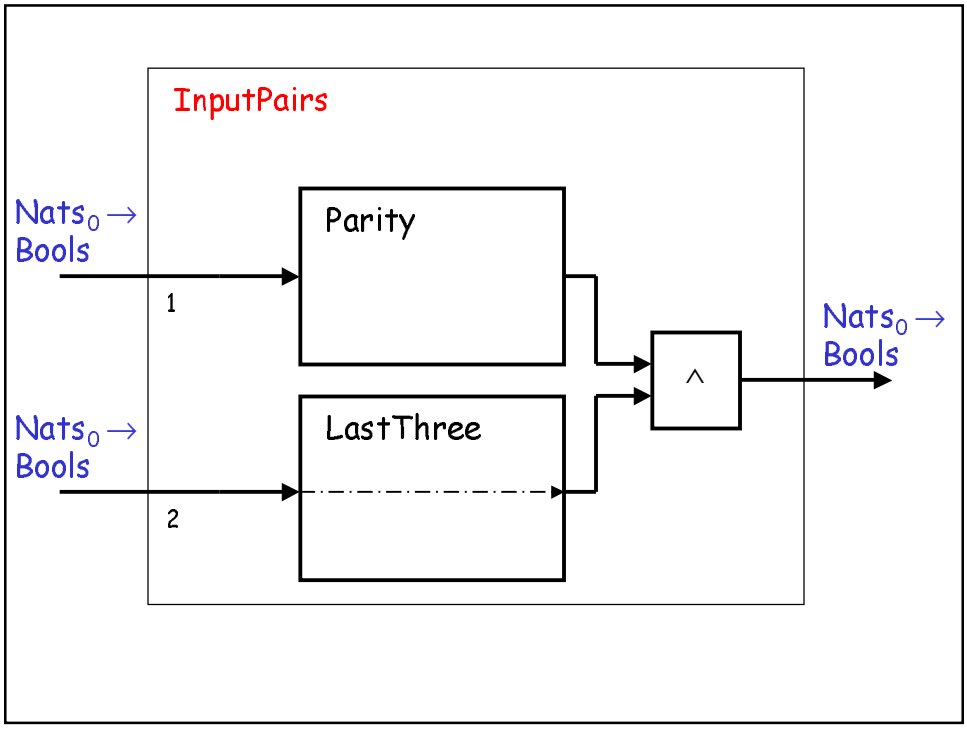
nextState [ParityAndLastThree] ((q1, q2) , x)

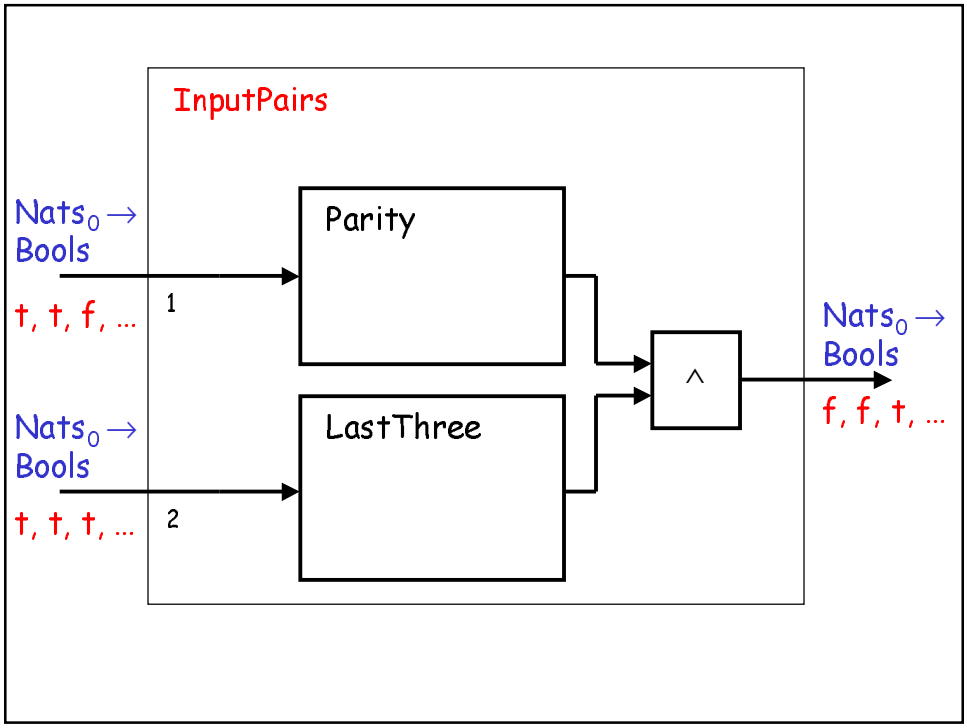
= (nextState [Parity] (q1, x) , nextState [LastThree] (q2, x))

output [ParityAndLastThree] ((q1, q2) , x)

= output [Parity] (q1, x) ∧ output [LastThree] (q2, x)




InputPairs
Nats$_0$ → Bools
1
Parity
Nats$_0$ → Bools
2
LastThree
^
Nats$_0$ → Bools


InputPairs
Nats$_0$ → Bools
t, t, f, ...
1
Parity
Nats$_0$ → Bools
t, t, t, ...
2
LastThree
^
Nats$_0$ → Bools
f, f, t, ...

The InputPairs System

Inputs [InputPairs] = Bools $\times$ Bools

$\mathit{Outputs}$ [InputPairs] = Bools

States [InputPairs]

= States [Parity] $\times$ States [LastThree]

= { true, false } $\times$ { 0, 1, 2 }

$\mathit{initialState}$ [InputPairs]

= (initialState [Parity], initialState [LastThree])

= (true, 0)

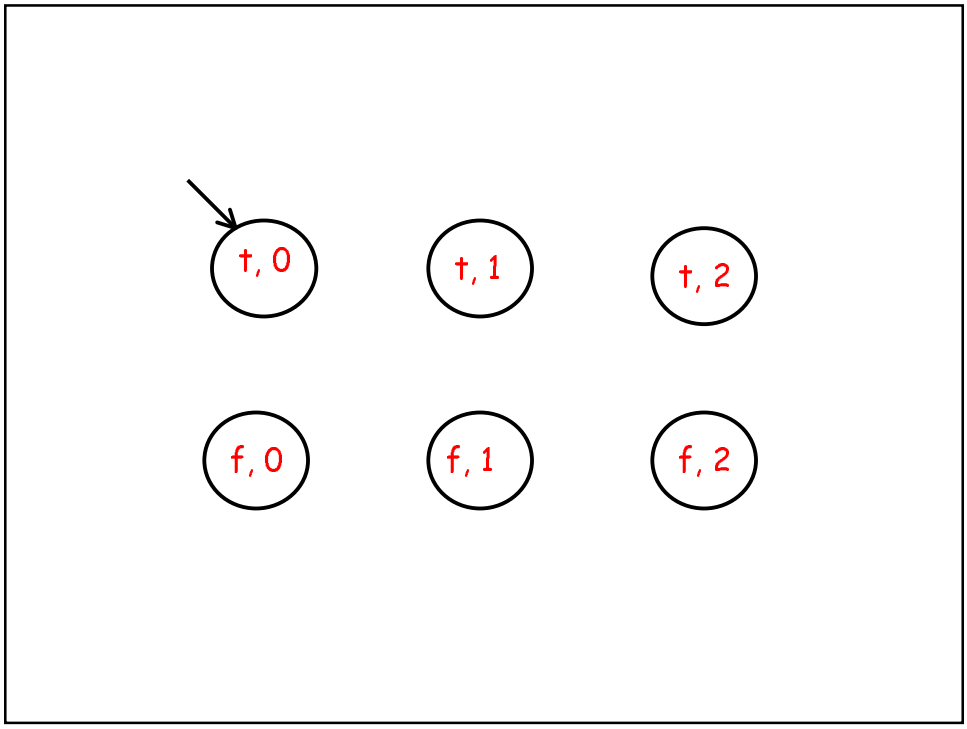
The InputPairs System, continued

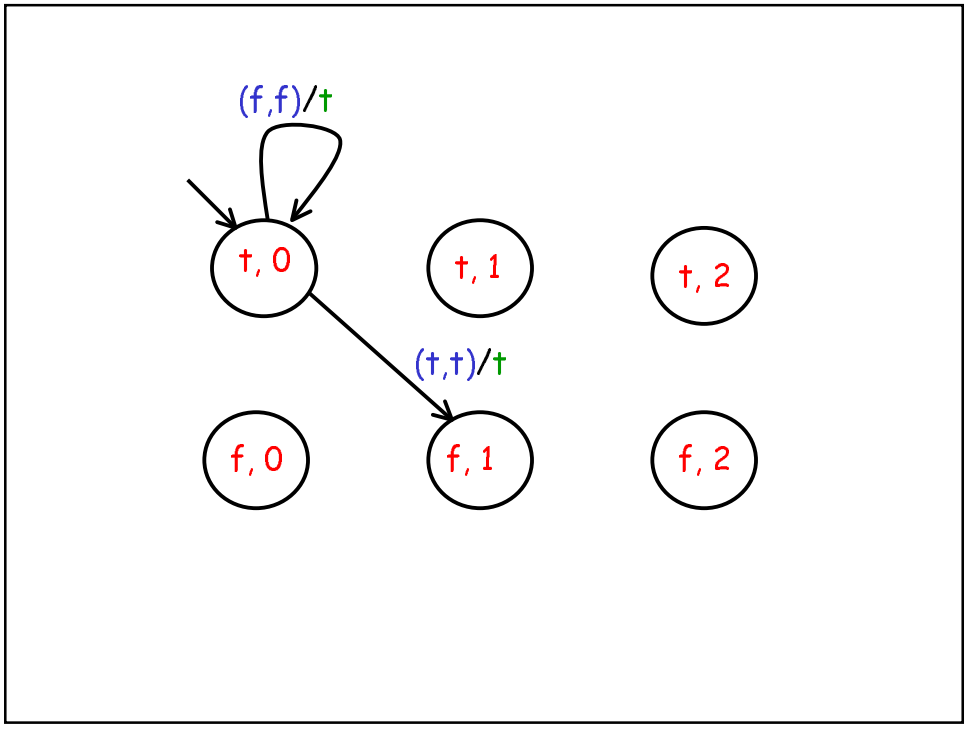
$\mathit{nextState}$ [InputPairs] ((q1, q2) , (x1, x2))

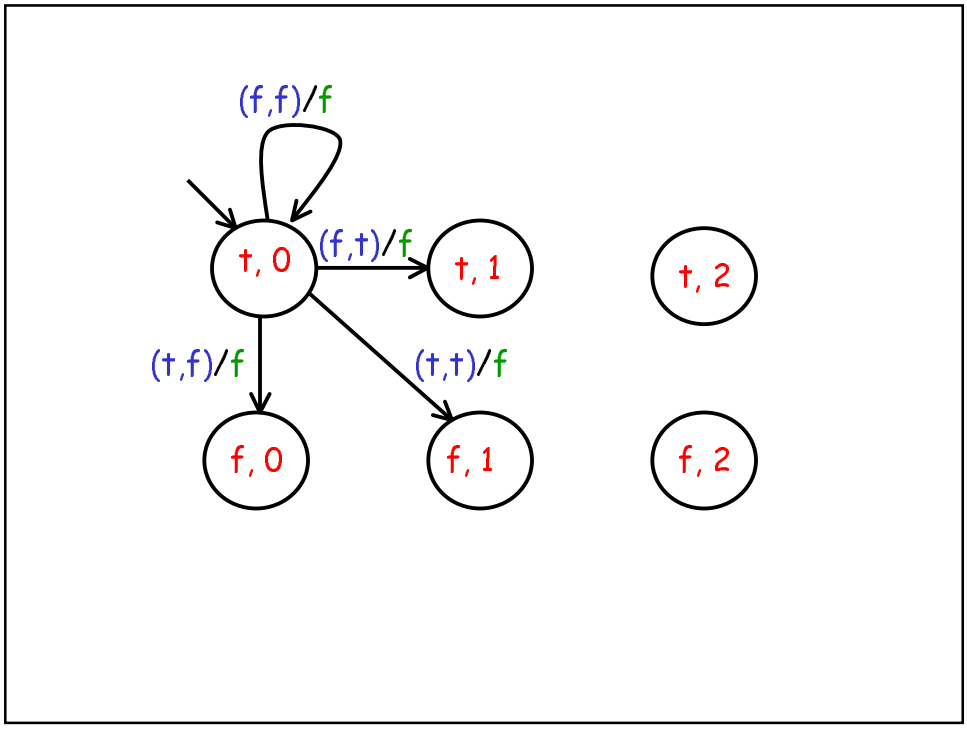
= (nextState [Parity] (q1, x1) , nextState [LastThree] (q2, x2))

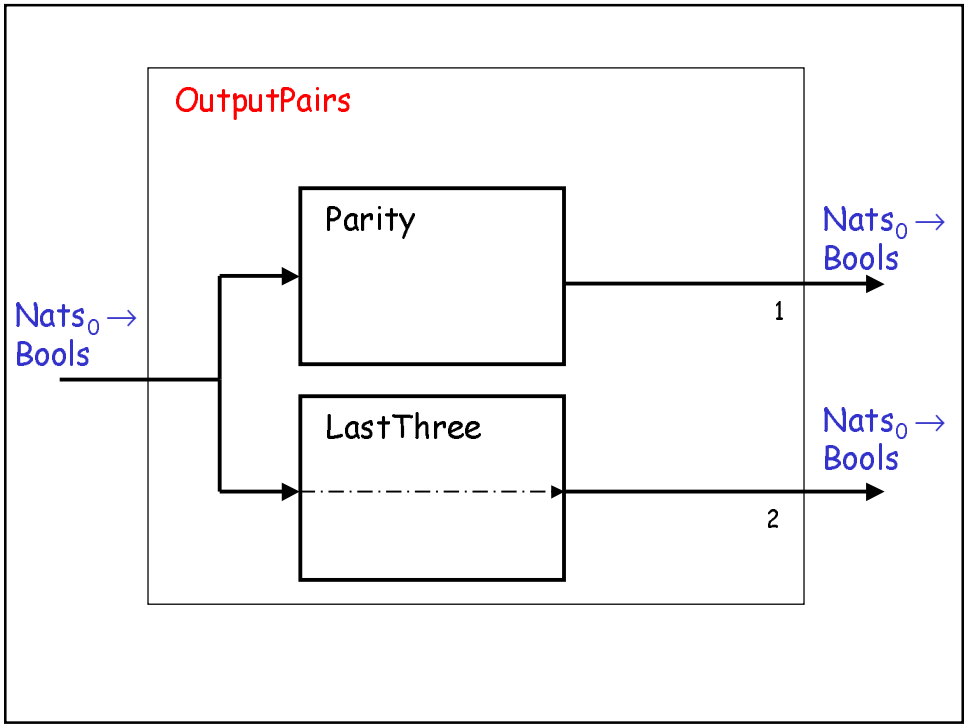
output [InputPairs] ((q1, q2) , (x1, x2))

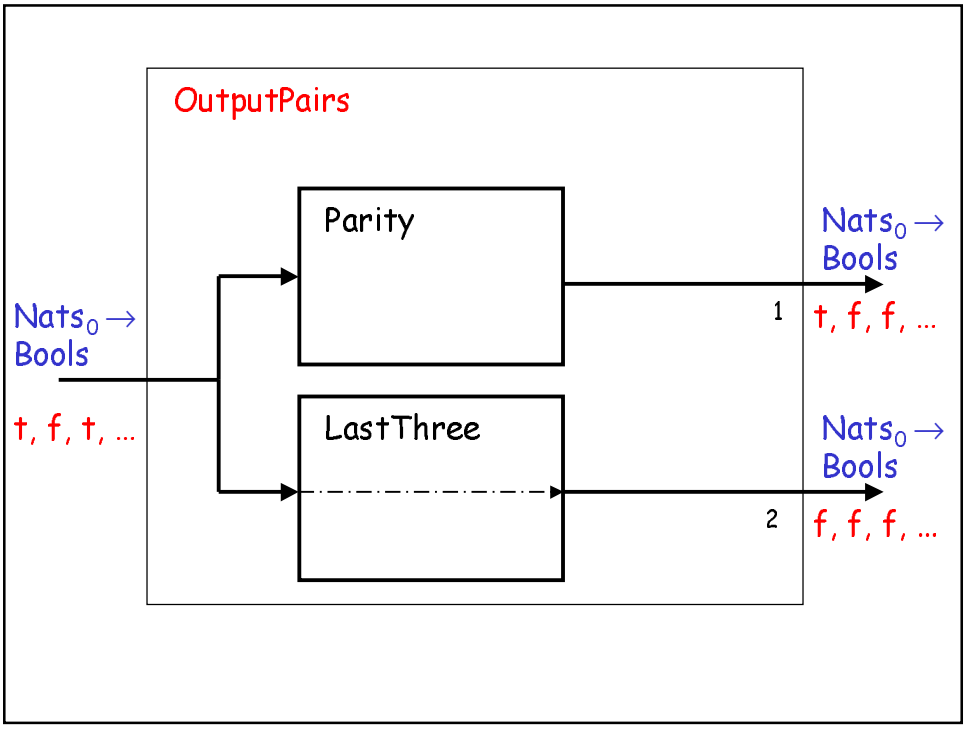
= output [Parity] (q1, x1) $\wedge$ output [LastThree] (q2, x2)


t, 0
t, 1
t, 2
f, 0
f, 1
f, 2


(f,f)/t
t, 0
t, 1
t, 2
(t,t)/t
f, 0
f, 1
f, 2


(f,f)/f
t, 0
(f,t)/f
t, 1
t, 2
(t,f)/f
(t,t)/f
f, 0
f, 1
f, 2


OutputPairs
Parity
LastThree
Nats$_0$ → Bools
Nats$_0$ → Bools
1
Nats$_0$ → Bools
2



The OutputPairs System

Inputs [OutputPairs] = Bools

Outputs [OutputPairs] = Bools × Bools

States [OutputPairs]

= States [Parity] × States [LastThree]

= { true, false } × { 0, 1, 2 }

initialState [OutputPairs]

= (initialState [Parity], initialState [LastThree])

= (true, 0)

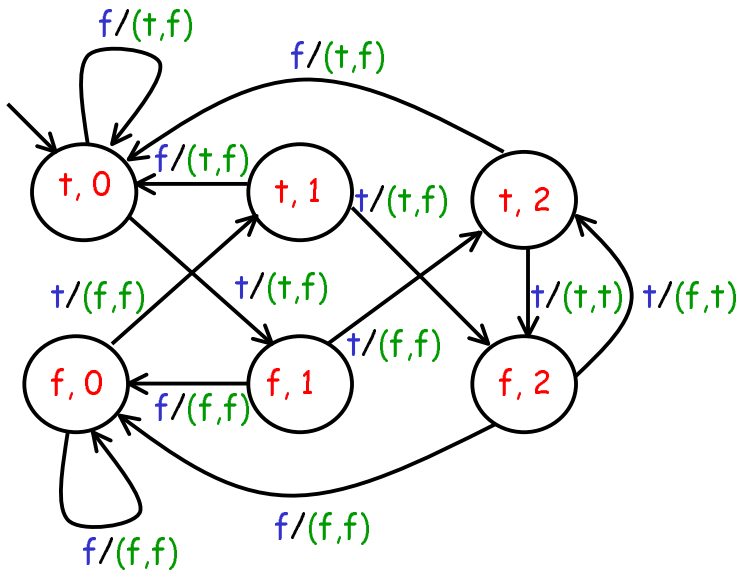
The OutputPairs System, continued

nextState [OutputPairs] ((q1, q2) , x)

= (nextState [Parity] (q1, x) , nextState [LastThree] (q2, x))

output [OutputPairs] ((q1, q2) , x)

= (output [Parity] (q1, x) , output [LastThree] (q2, x))



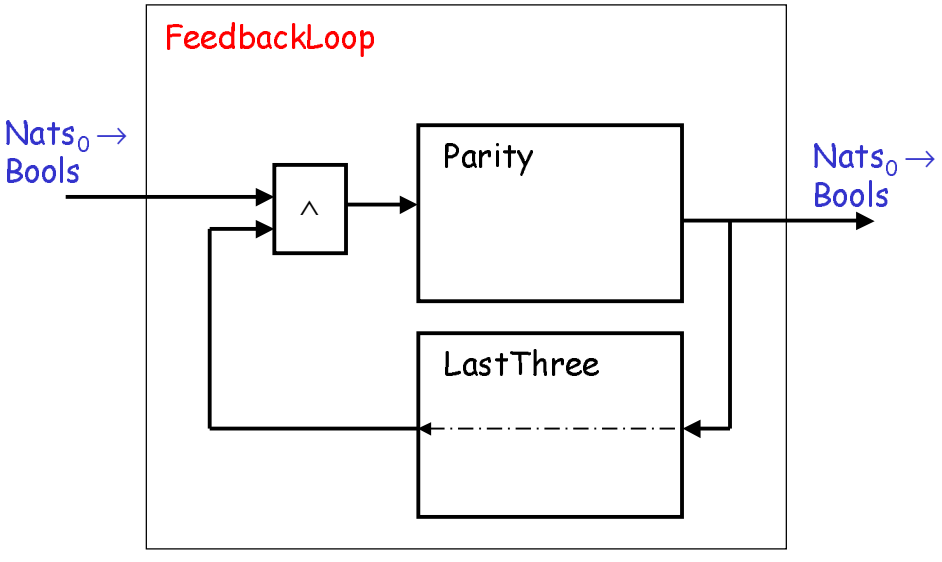
Any block diagram of N state machines with the state spaces

States1, States2, … StatesN

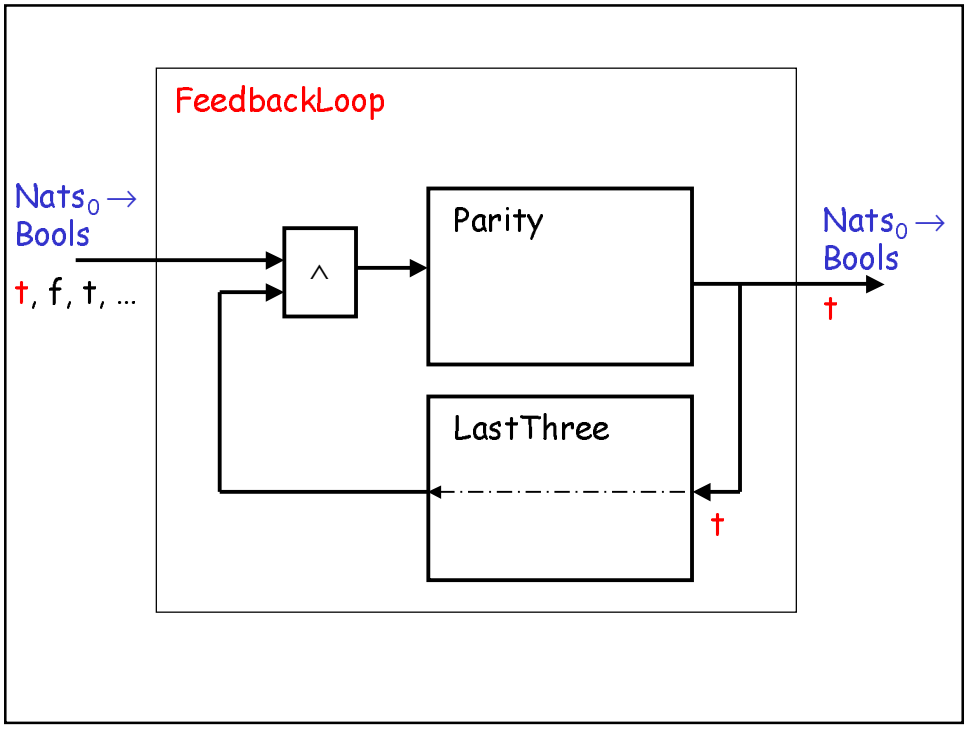
can be implemented by a single state machine with the state space

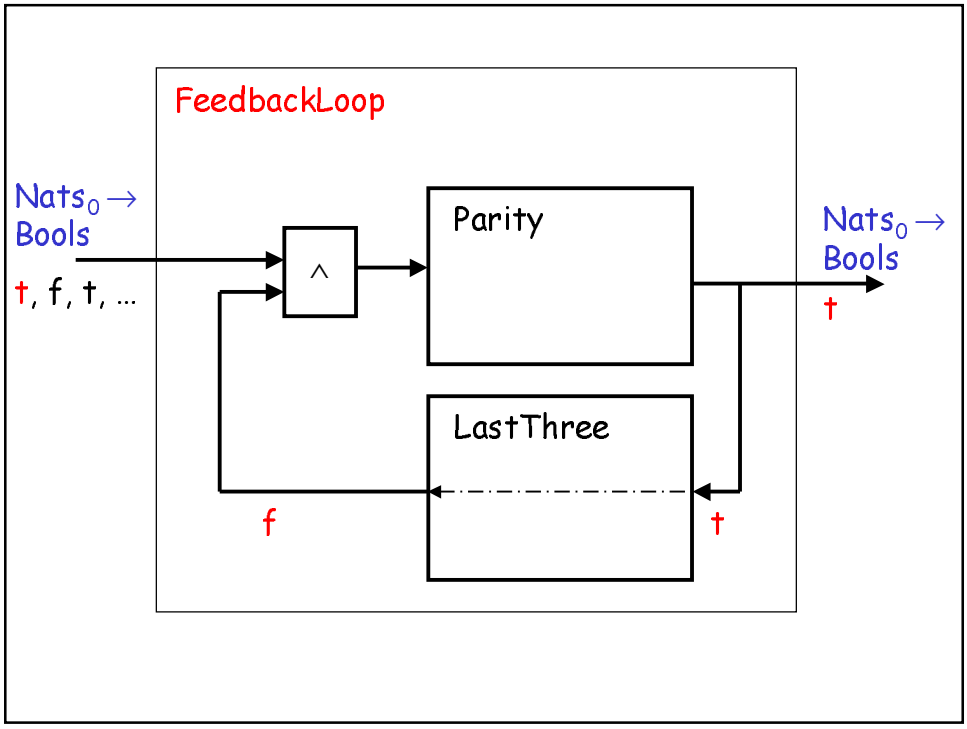
States1 × States2 × … × StatesN .

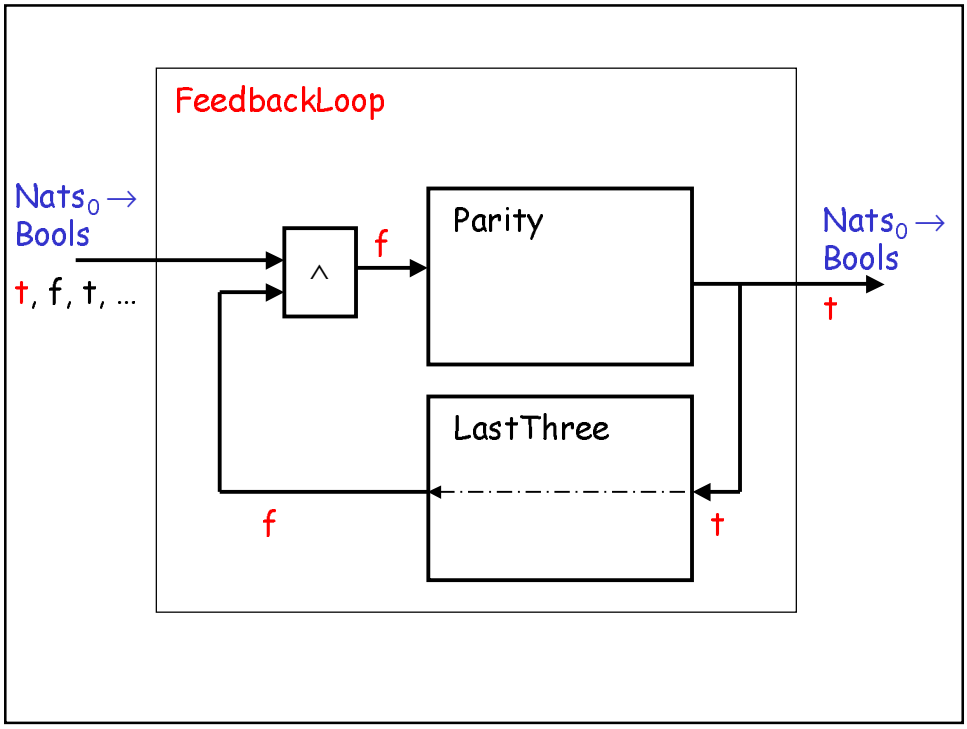
This is called a "product machine".

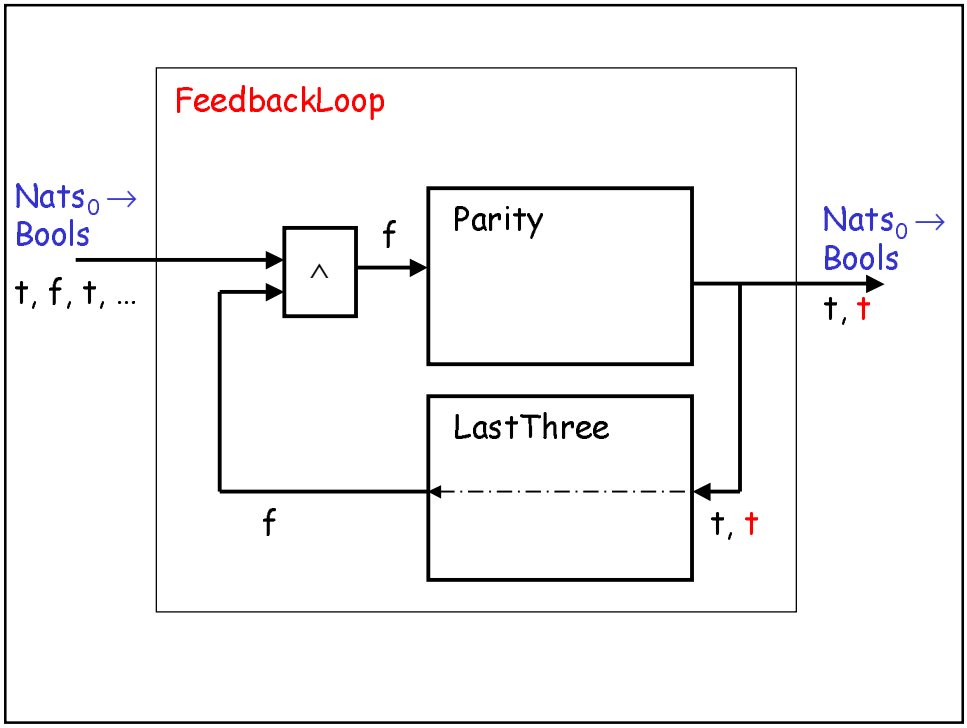


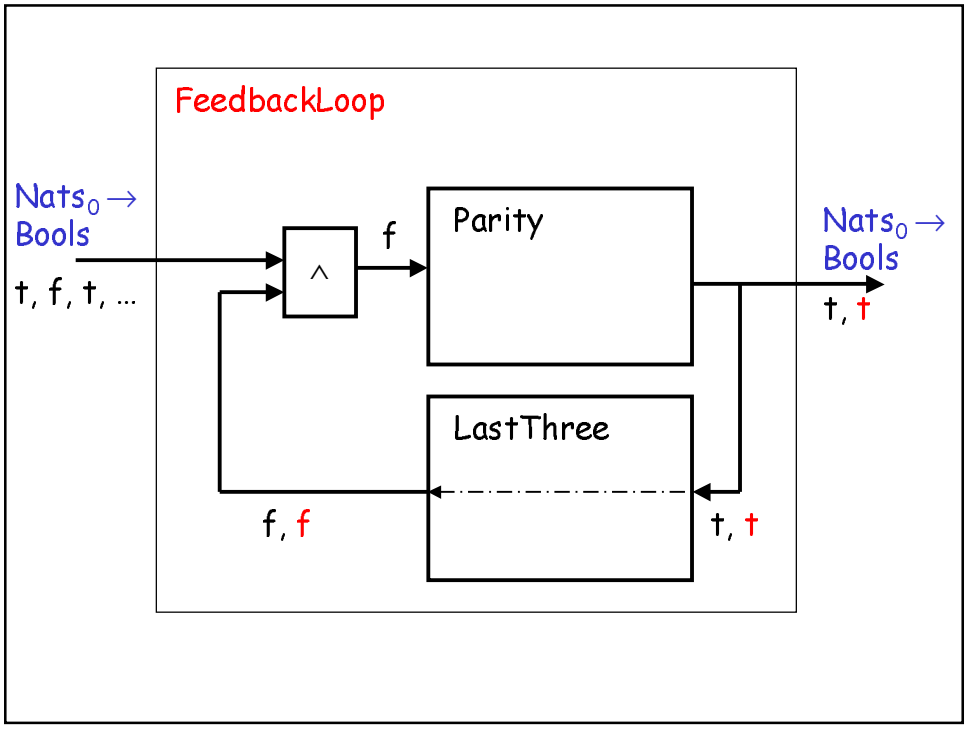
This block diagram is ok, because every cycle contains a delay.

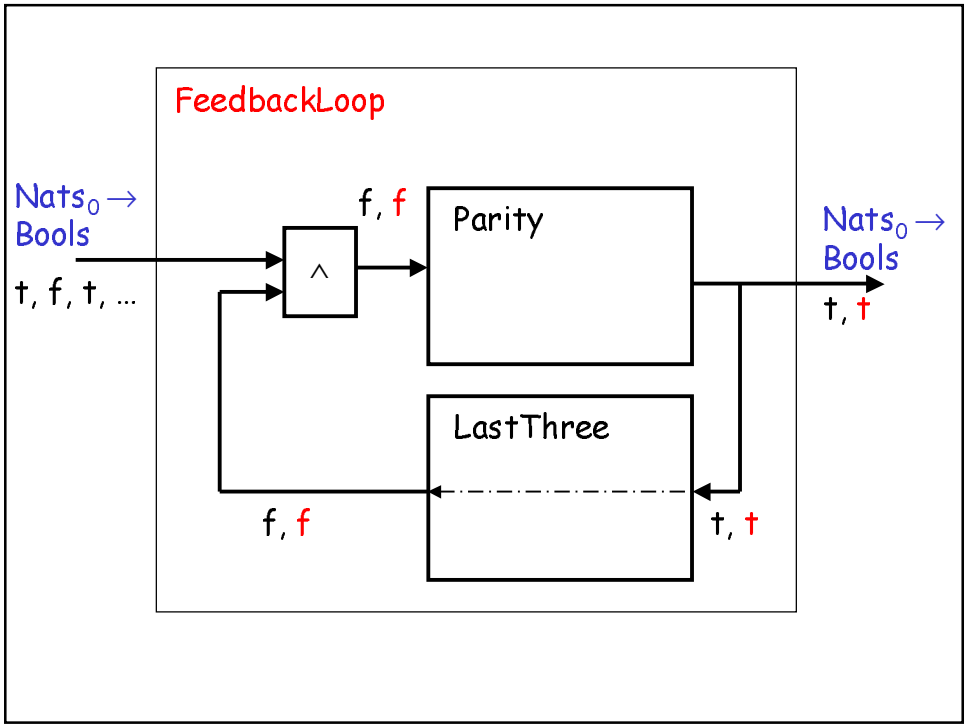

FeedbackLoop
Nats$_0$ → Bools
t, f, t, ...
^
Parity
LastThree
Nats$_0$ → Bools
t
t

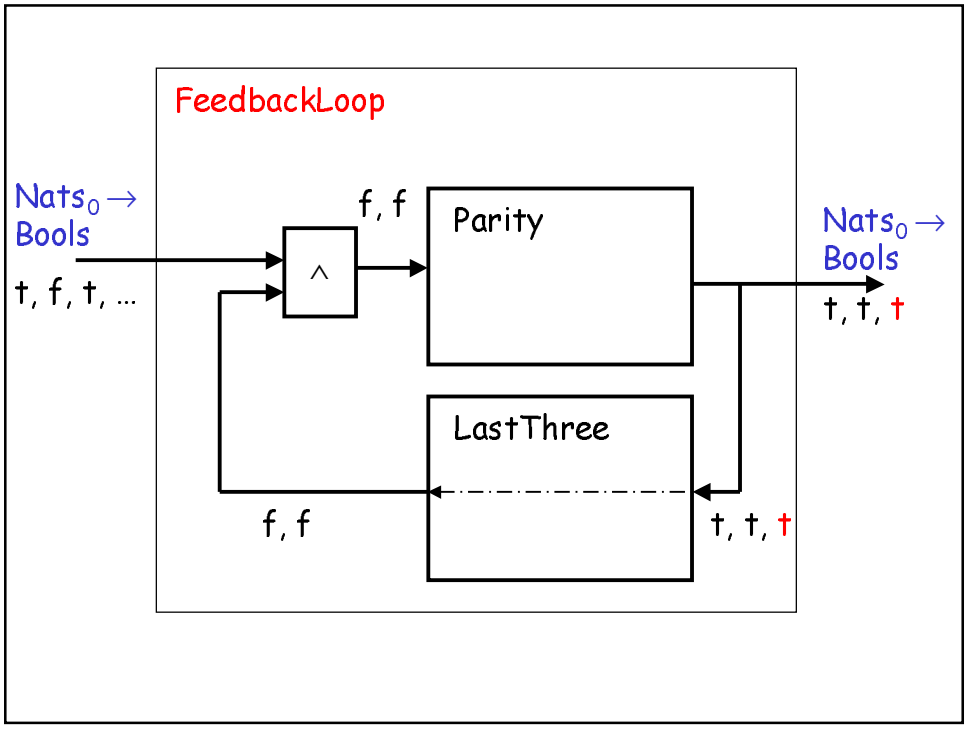

FeedbackLoop
Nats$_0$ → Bools
t, f, t, ...
^
Parity
LastThree
Nats$_0$ → Bools
t
f
t

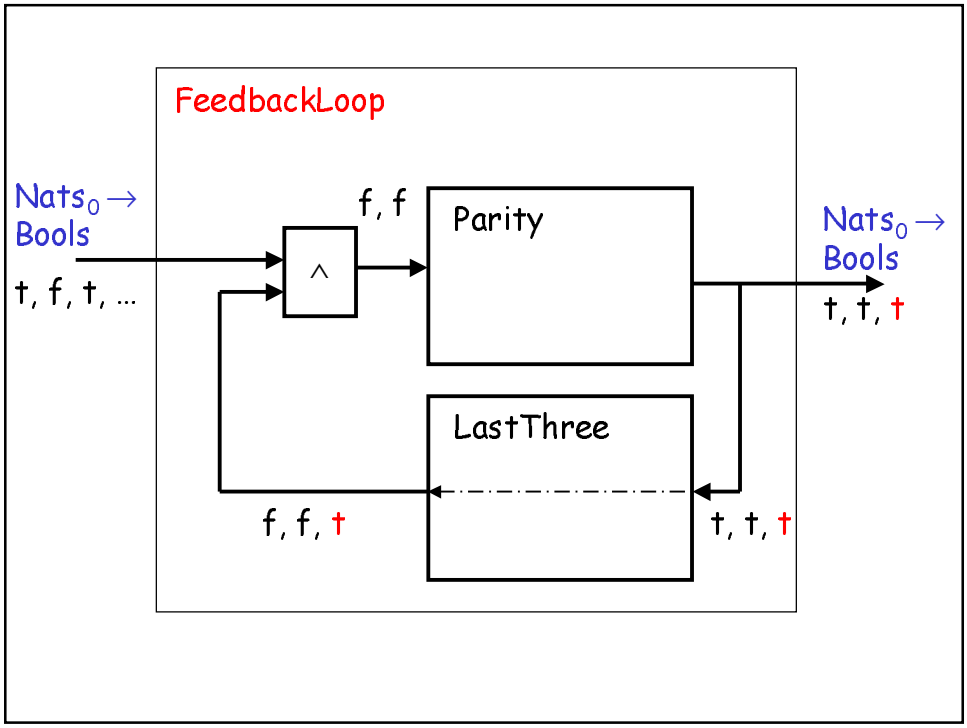

FeedbackLoop
Nats$_0$ → Bools
t, f, t, …
^
f
Parity
Nats$_0$ → Bools
t
LastThree
f
t

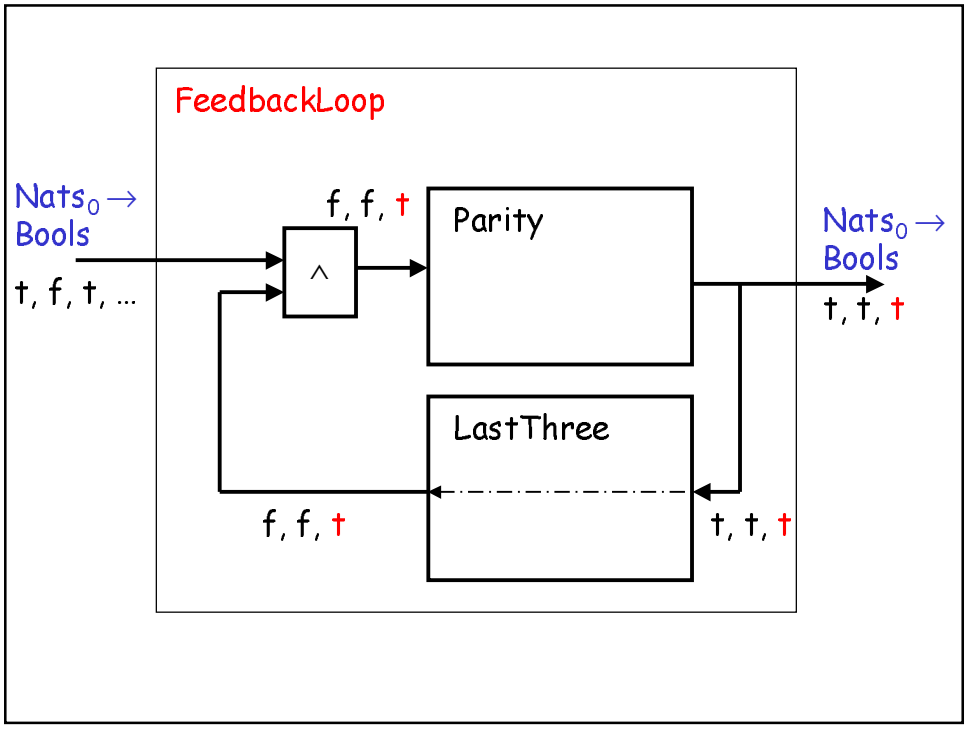

FeedbackLoop
Nats$_0$ → Bools
t, f, t, …
^
f
Parity
Nats$_0$ → Bools
t, t
LastThree
f
t, t

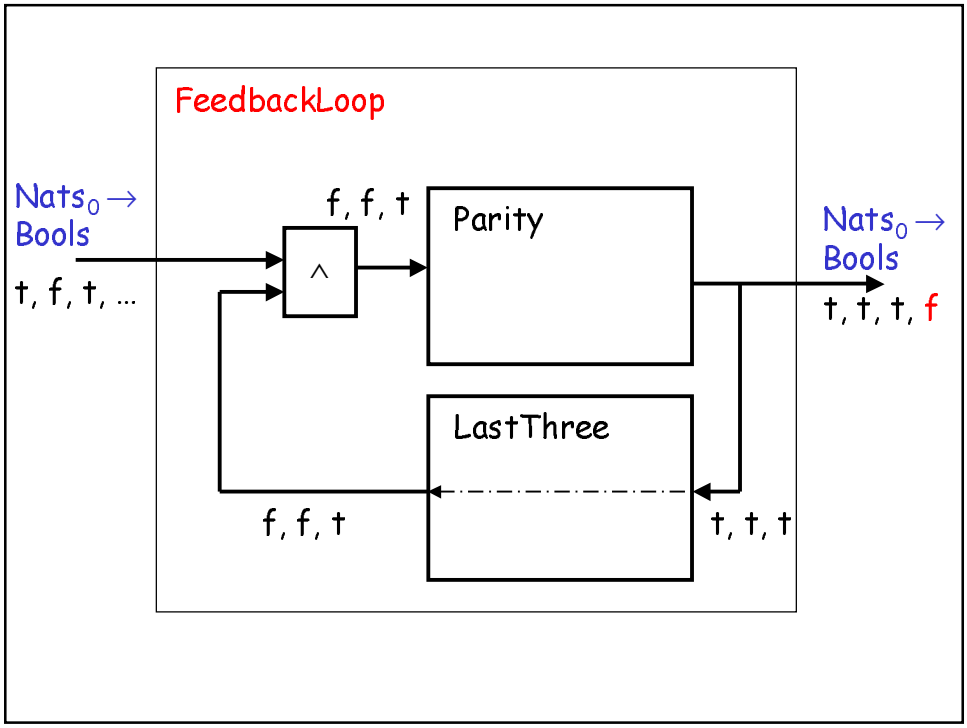

FeedbackLoop
Nats$_0$ → Bools
t, f, t, ...
^
f
Parity
Nats$_0$ → Bools
t, t
LastThree
f, f
t, t


FeedbackLoop
Nats$_0$ → Bools
t, f, t, ...
f, f
^
Parity
Nats$_0$ → Bools
t, t
LastThree
f, f
t, t


FeedbackLoop
Nats$_0$ → Bools
t, f, t, ...
^
f, f
Parity
Nats$_0$ → Bools
t, t, t
LastThree
f, f
t, t, t

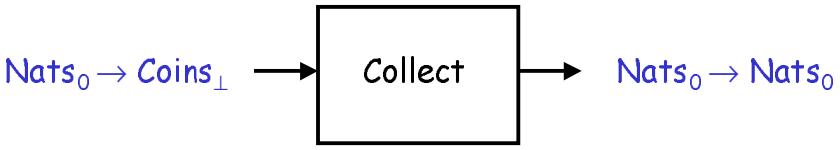

FeedbackLoop
Nats$_0$ → Bools
t, f, t, ...
^
f, f
Parity
Nats$_0$ → Bools
t, t, t
LastThree
f, f, t
t, t, t


FeedbackLoop
Nats0 → Bools
t, f, t, …
f, f, t
^
Parity
Nats0 → Bools
t, t, t
LastThree
f, f, t
t, t, t


FeedbackLoop
Nats0 → Bools
t, f, t, …
f, f, t
^
Parity
Nats0 → Bools
t, t, t, f
LastThree
f, f, t
t, t, t

Example: Vending Machine

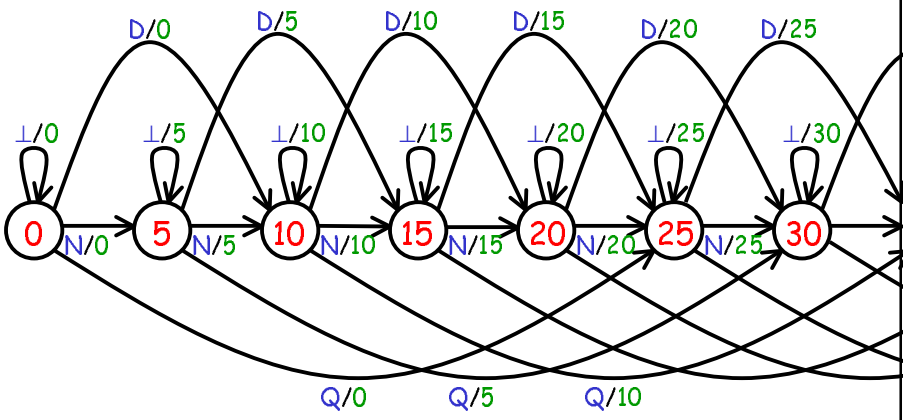
Coin Collector



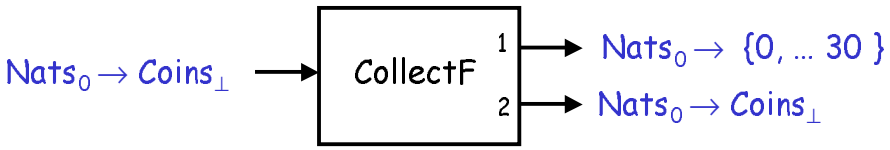
Let Coins = { Nickel, Dime, Quarter } .

Let $Coins_\perp$ = Coins $\cup$ { $\perp$ } . ($\perp$ stands for "no input.")

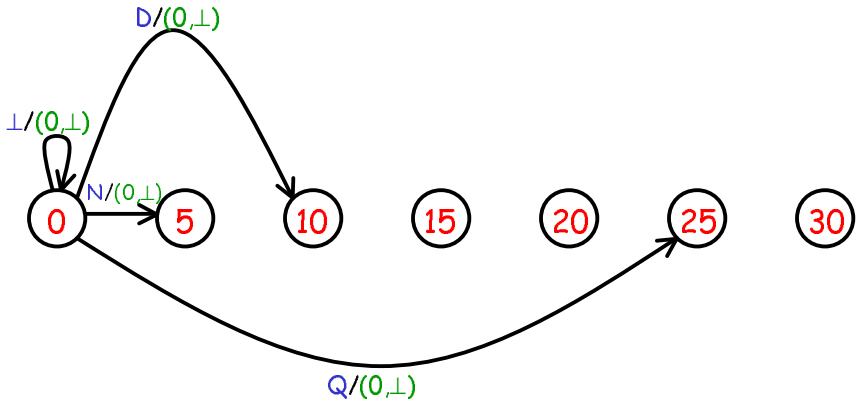
Coin Collector



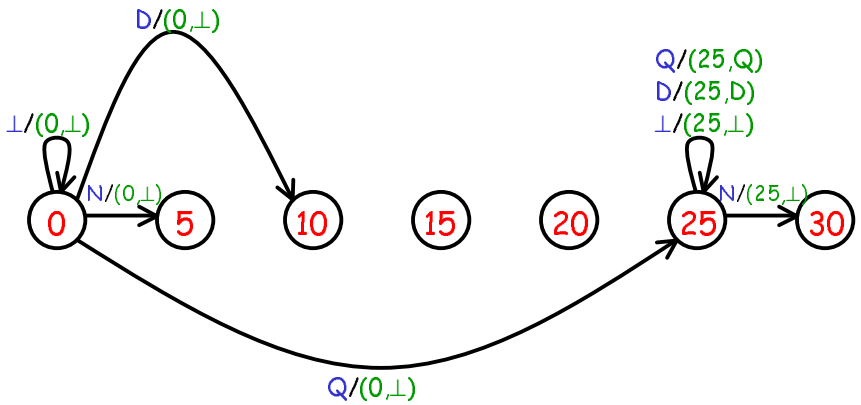
Finite-State Coin Collector



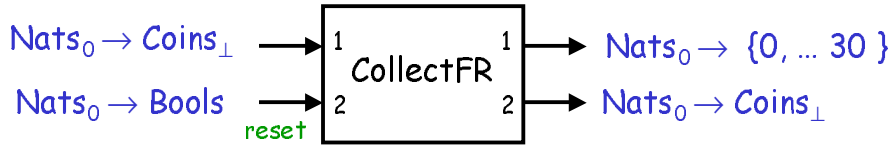
Finite-State Coin Collector



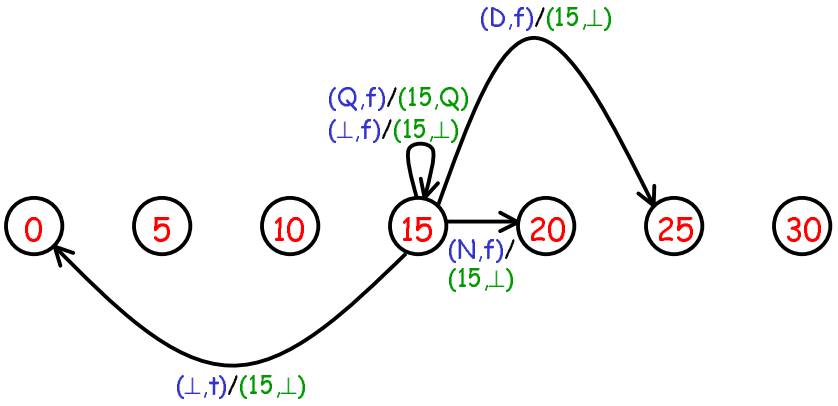
Finite-State Coin Collector



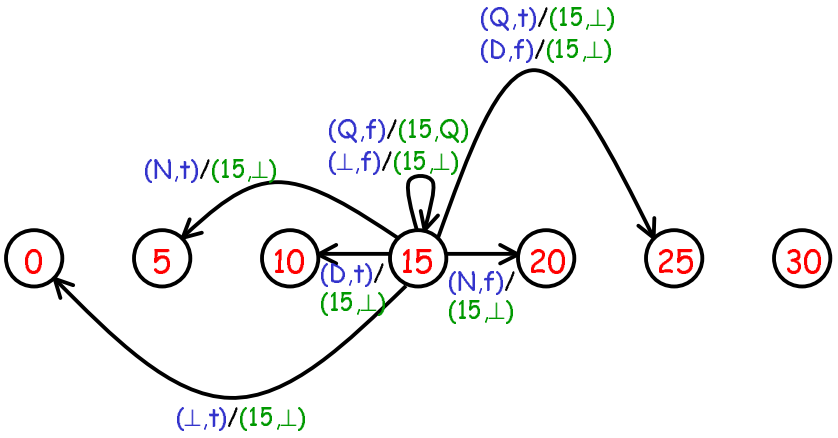
Coin Collector with Reset



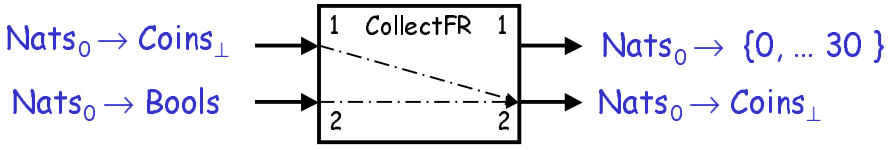
Coin Collector with Reset



Coin Collector with Reset



Coin Collector with Reset

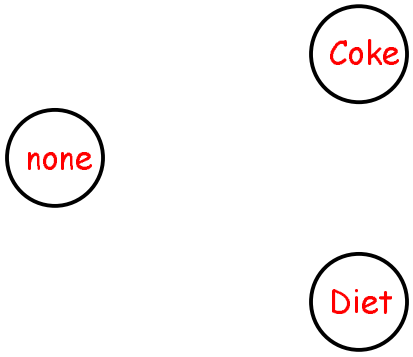


Soda Dispenser

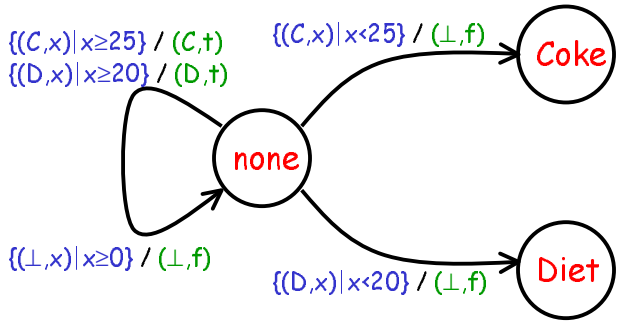
Let Select = { selectCoke, selectDiet } .

Let Dispense = { dispenseCoke, dispenseDiet } .

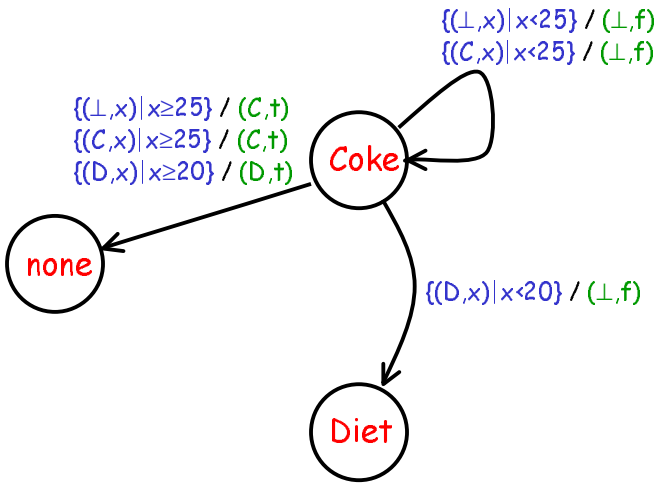
Soda Dispenser



Soda Dispenser

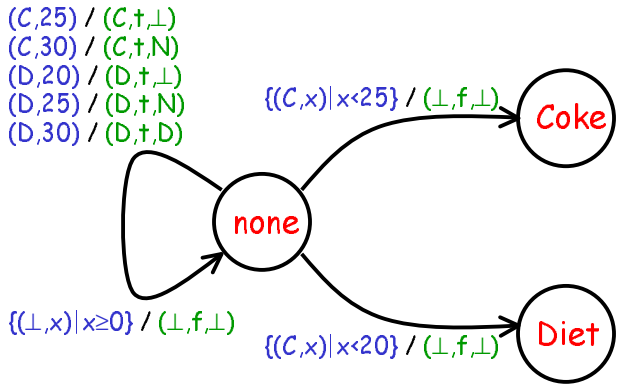


Soda Dispenser

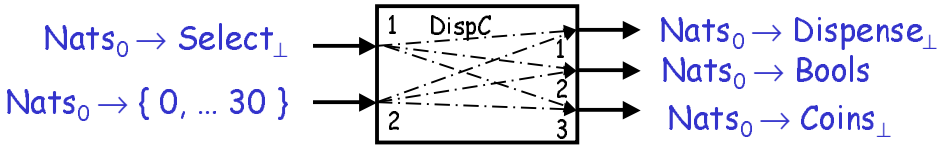


Soda Dispenser with Change

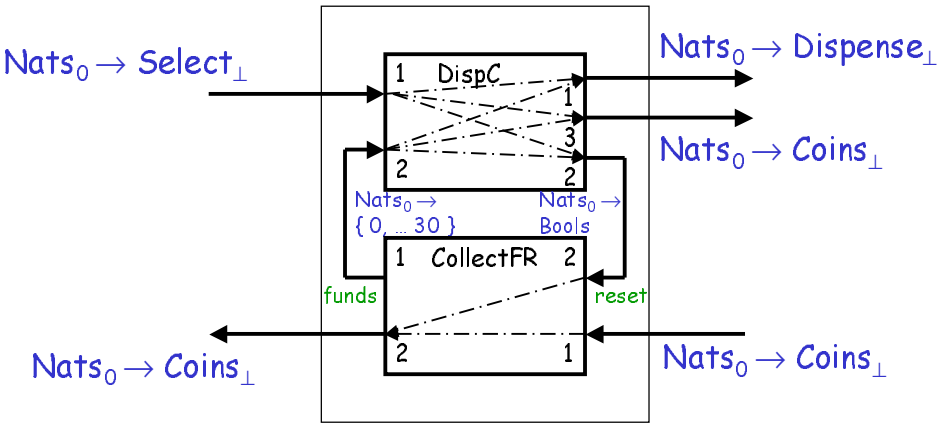
Soda Dispenser with Change



Soda Dispenser with Change



Vending Machine



State Space of Vending Machine

$\{ 0, 5, 10, 15, 20, 25, 30 \} \times \{ \text{none}, \text{Coke}, \text{Diet} \}$

21 states