State Machines

EECS 20

Lecture 8 (February 2, 2001)

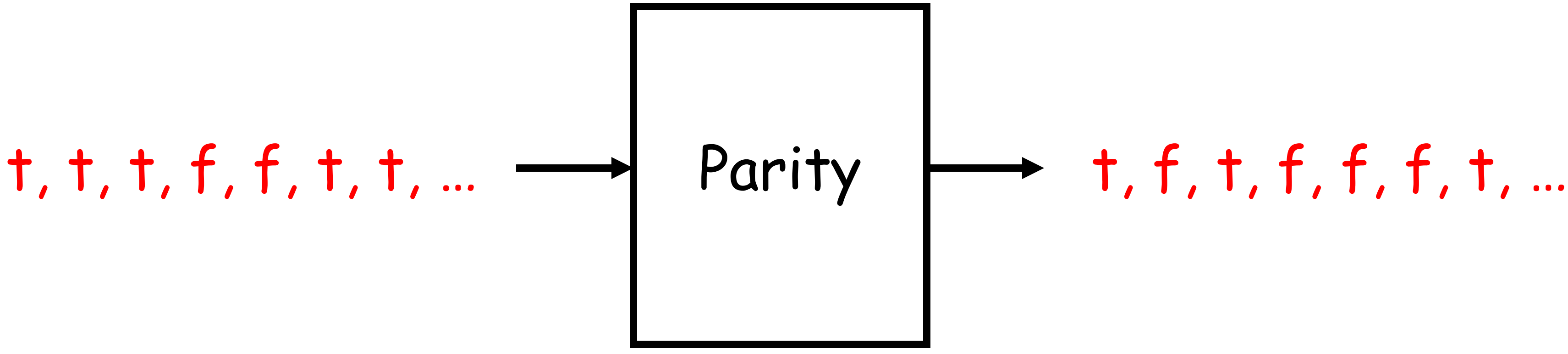
Tom Henzinger

Finite-Memory Systems

Discrete-Time Delay:	remember last input value
Discrete-Time Moving Average:	remember last 2 input values
Parity:	remember if number of past inputs "true" is even

Infinite-Memory Systems

Count:	remember if number of past inputs "true" is greater than number of past inputs "false"


t, t, t, f, f, t, t, ...
Parity
t, f, t, f, f, f, t, ...

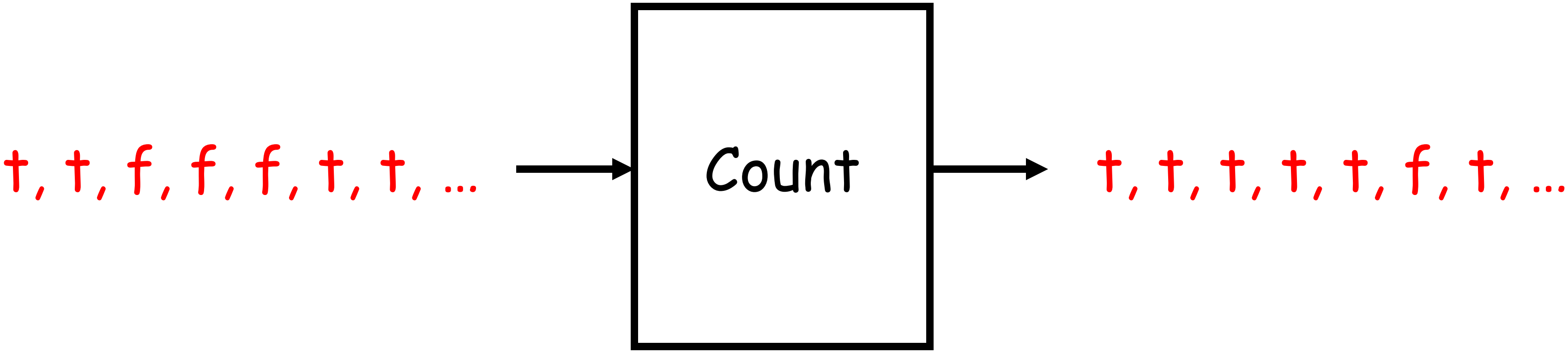
The Parity System

$$\text{Parity} : [\, \text{Nats}_0 \to \text{Bools} \,] \to [\, \text{Nats}_0 \to \text{Bools} \,]$$

such that $\forall\, x \in [\, \text{Nats}_0 \to \text{Bools} \,]$, $\forall\, y \in \text{Nats}_0$,

$$(\,\text{Parity}\,(x)\,)\,(y) = \begin{cases} \text{true} & \text{if } |\,\text{trueValues}\,(x,y)\,| \text{ is even} \\ \text{false} & \text{if } |\,\text{trueValues}\,(x,y)\,| \text{ is odd} \end{cases}$$

where $\text{trueValues}\,(x,y) = \{\, z \in \text{Nats}_0 \mid z < y \wedge x\,(z) = \text{true} \,\}$


t, t, f, f, f, t, t, ...
Count
t, t, t, t, t, f, t, ...

The Count System

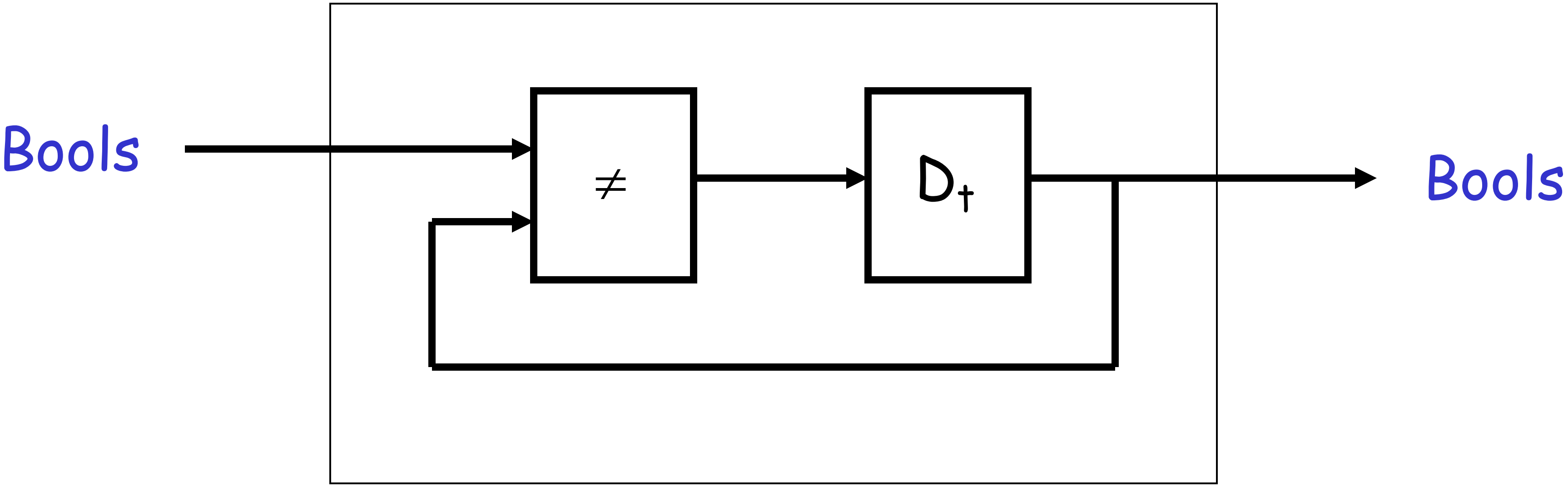
$$\text{Count} : [\,\text{Nats}_0 \rightarrow \text{Bools}\,] \rightarrow [\,\text{Nats}_0 \rightarrow \text{Bools}\,]$$

such that $\forall\, x \in [\,\text{Nats}_0 \rightarrow \text{Bools}\,]$, $\forall\, y \in \text{Nats}_0$,

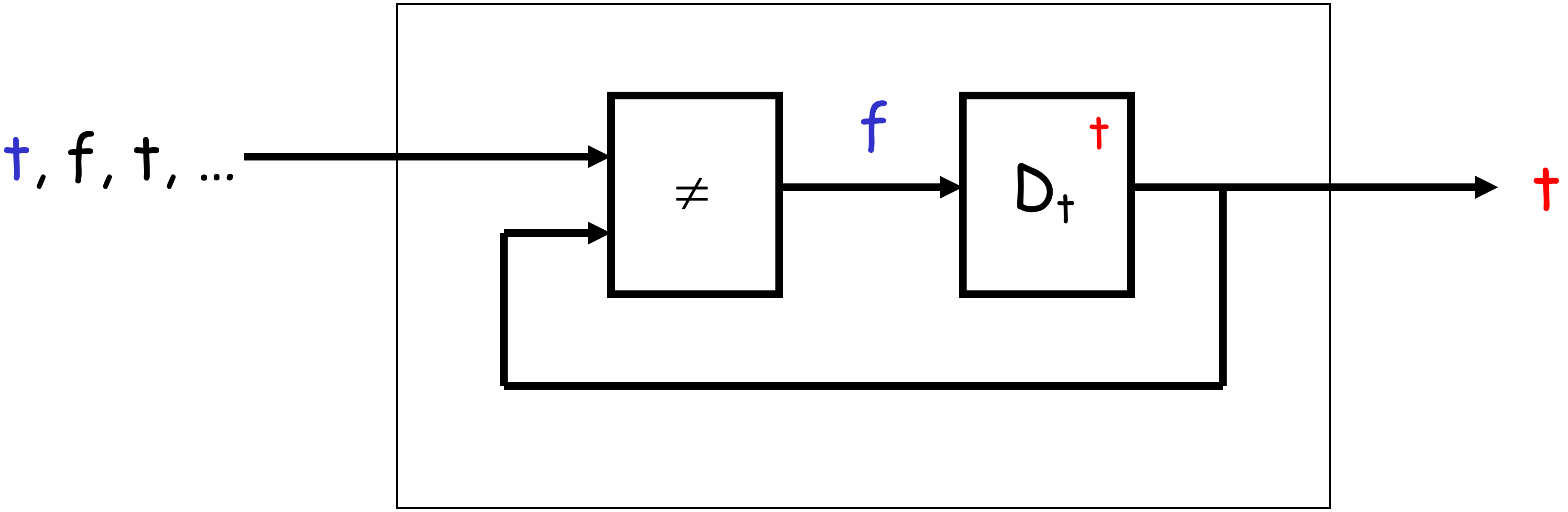
$$(\,\text{Count}\,(x)\,)\,(y) = \begin{cases} \text{true} & \text{if } |\,\text{trueValues}\,(x,y)\,| \geq |\,\text{falseValues}\,(x,y)\,| \\ \text{false} & \text{if } |\,\text{trueValues}\,(x,y)\,| < |\,\text{falseValues}\,(x,y)\,| \end{cases}$$

where $\text{falseValues}\,(x,y) = \{\, z \in \text{Nats}_0 \mid z < y \wedge x\,(z) = \text{false} \,\}$

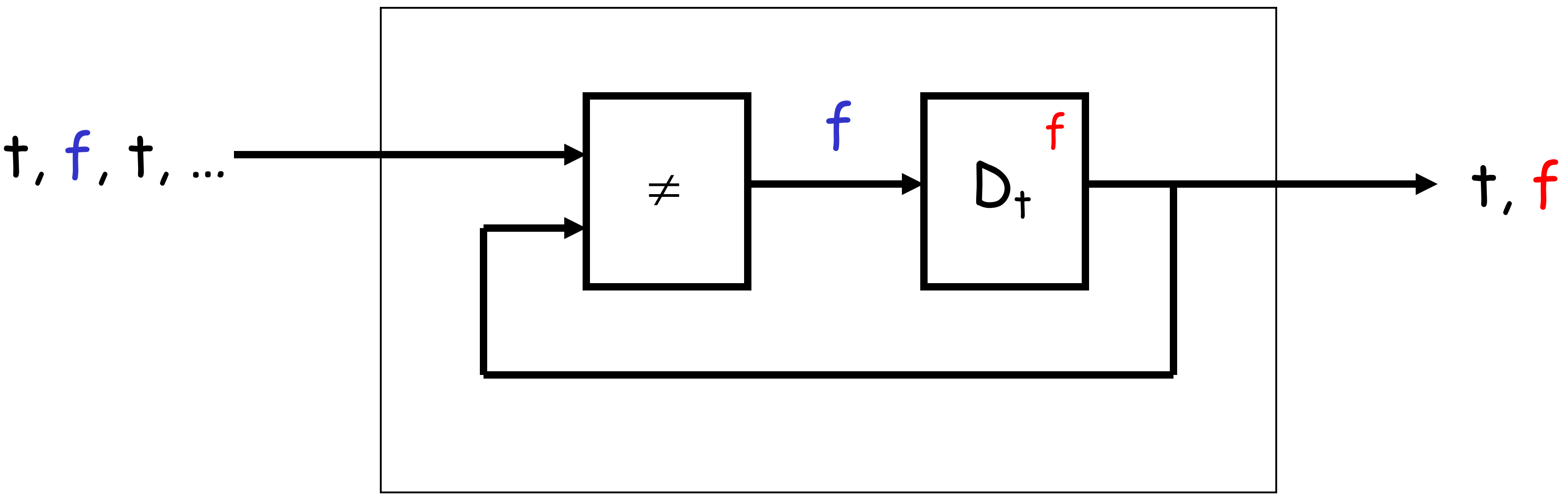
An Implementation of the Parity System



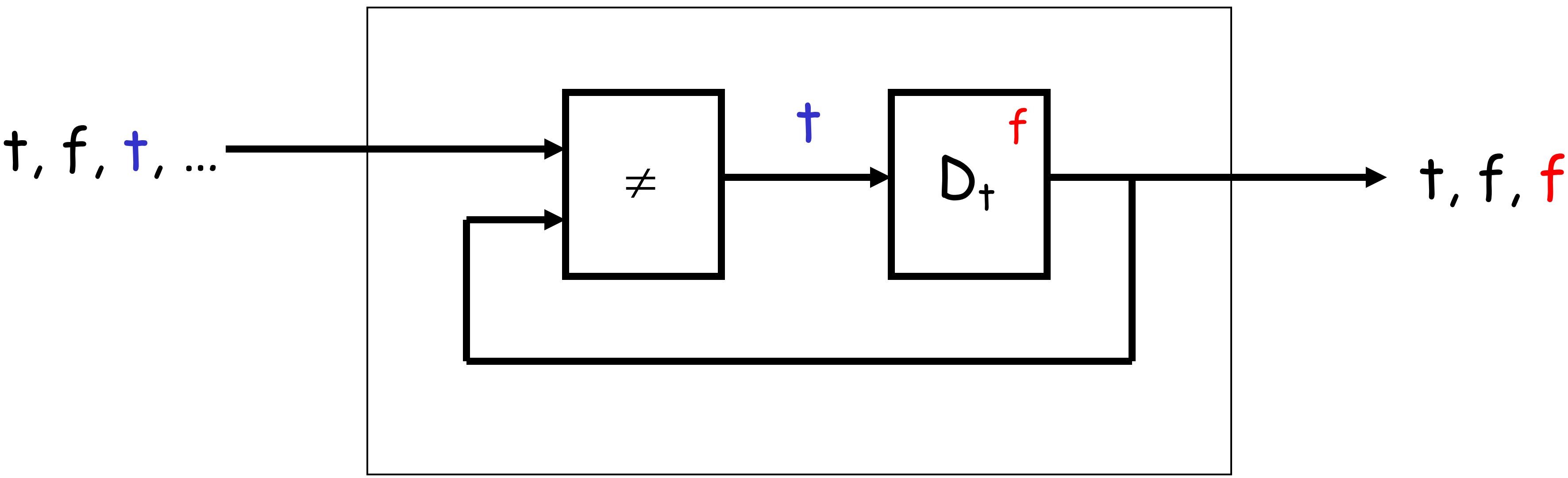
An Implementation of the Parity System



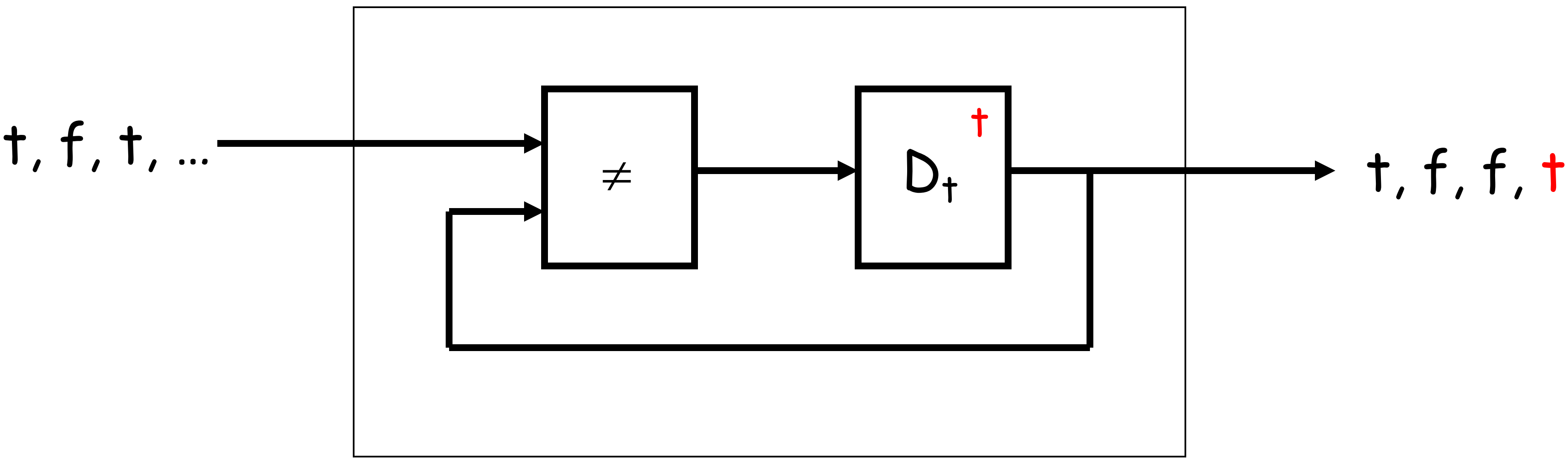
An Implementation of the Parity System



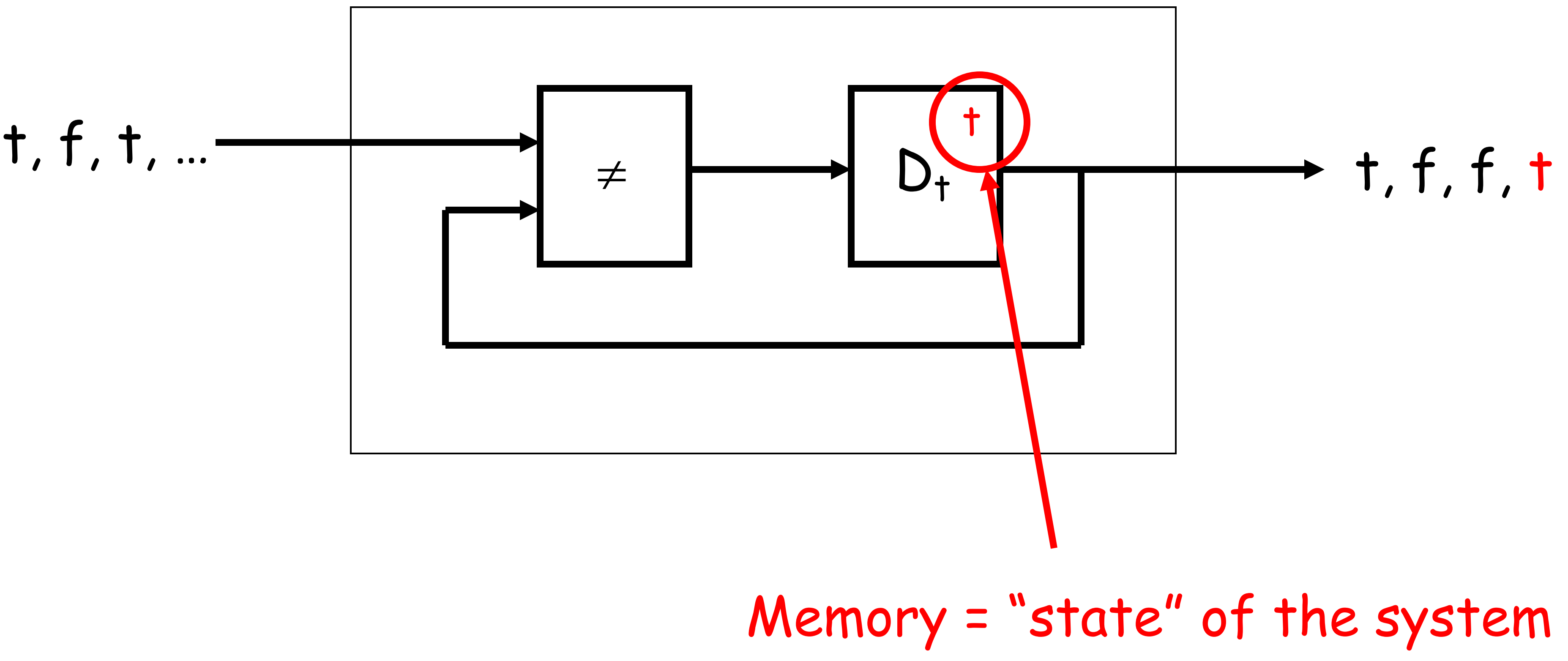
An Implementation of the Parity System



An Implementation of the Parity System



An Implementation of the Parity System



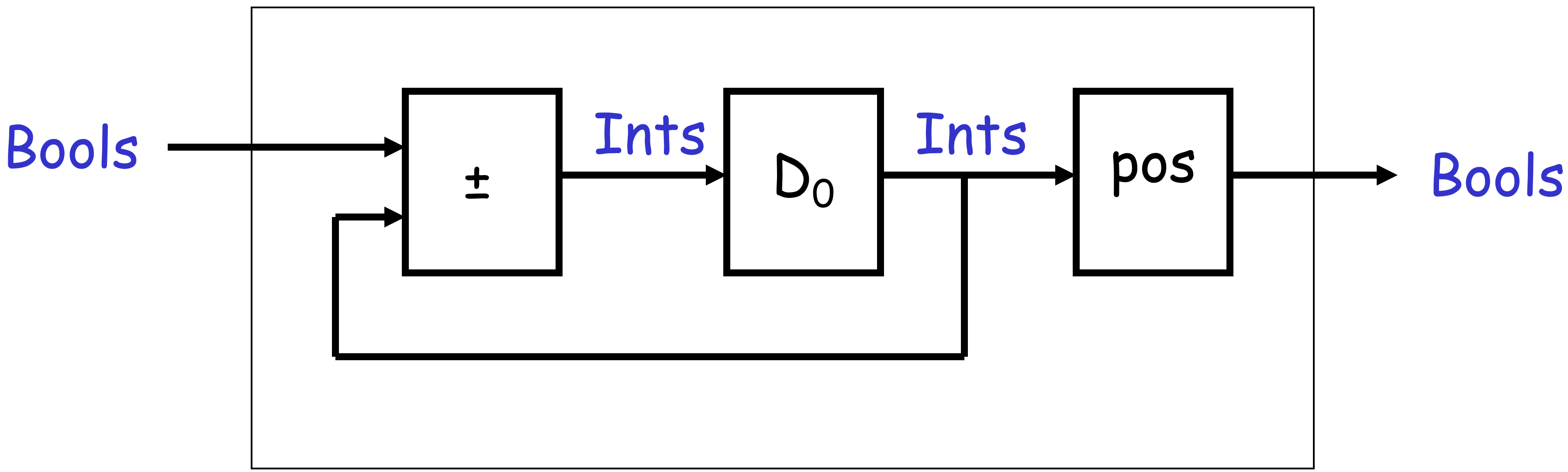
Systems with finite memory

are naturally implemented as

finite state machines

(or finite transition systems) .

An Implementation of the Count System



$\pm : \text{Bools} \times \text{Ints} \rightarrow \text{Ints}$

such that $\forall x \in \text{Bools}, \forall y \in \text{Ints},$

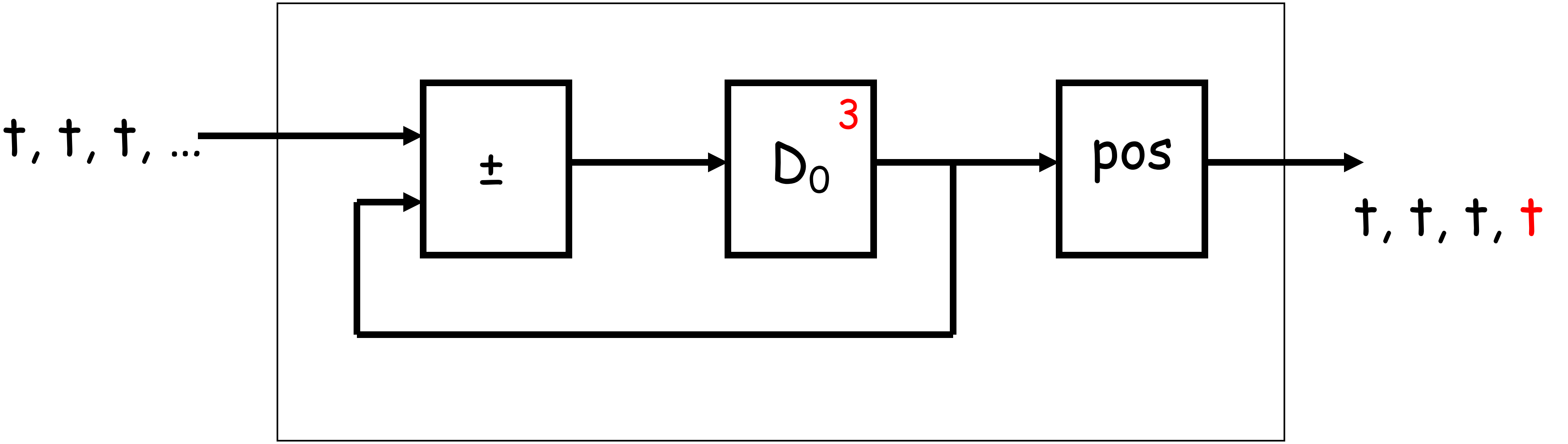
$$\pm(x,y) = \begin{cases} y+1 & \text{if } x = \text{true} \\ y-1 & \text{if } x = \text{false} \end{cases}$$

$\text{pos} : \text{Ints} \rightarrow \text{Bools}$

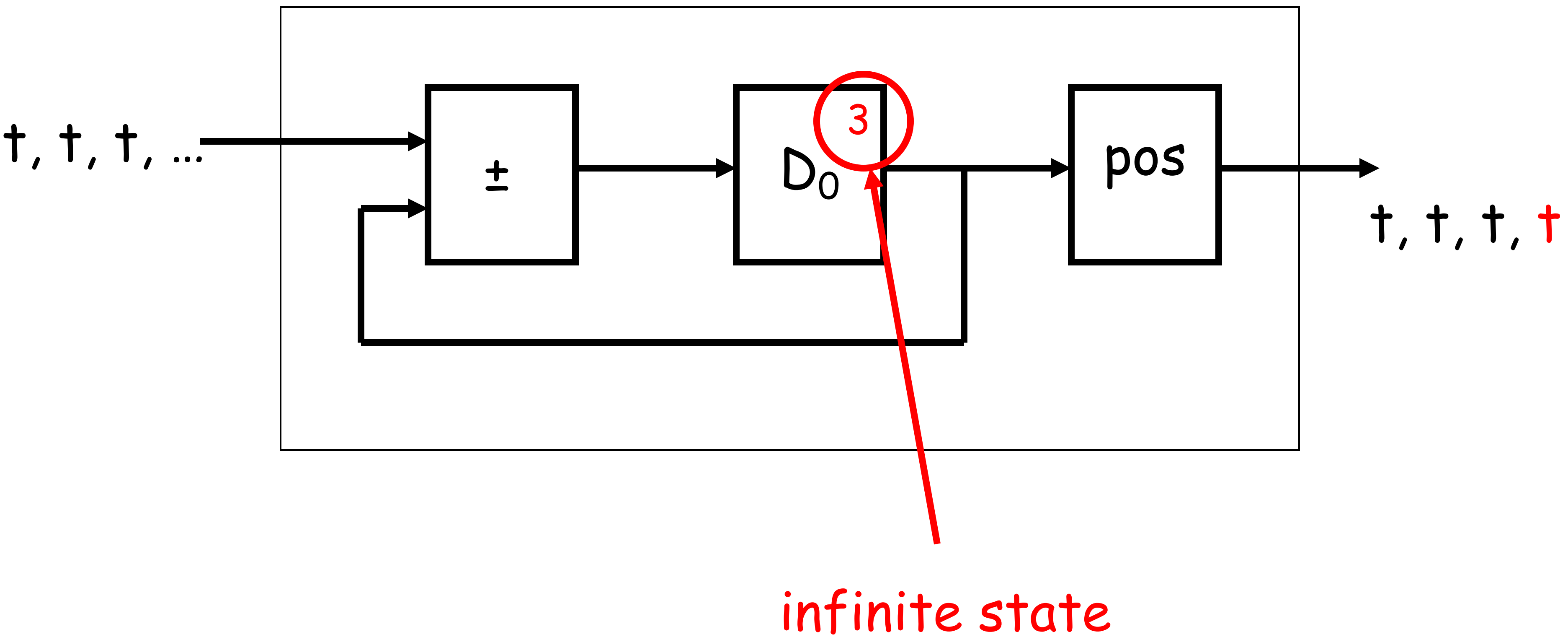
such that $\forall x \in \text{Ints},$

$$\text{pos}(x) = \begin{cases} \text{true} & \text{if } x \geq 0 \\ \text{false} & \text{if } x < 0 \end{cases}$$

An Implementation of the Count System



An Implementation of the Count System



Systems with countable (integer) memory

are naturally implemented as

infinite state machines

(or infinite transition systems) .

(Systems with uncountable (real) memory,

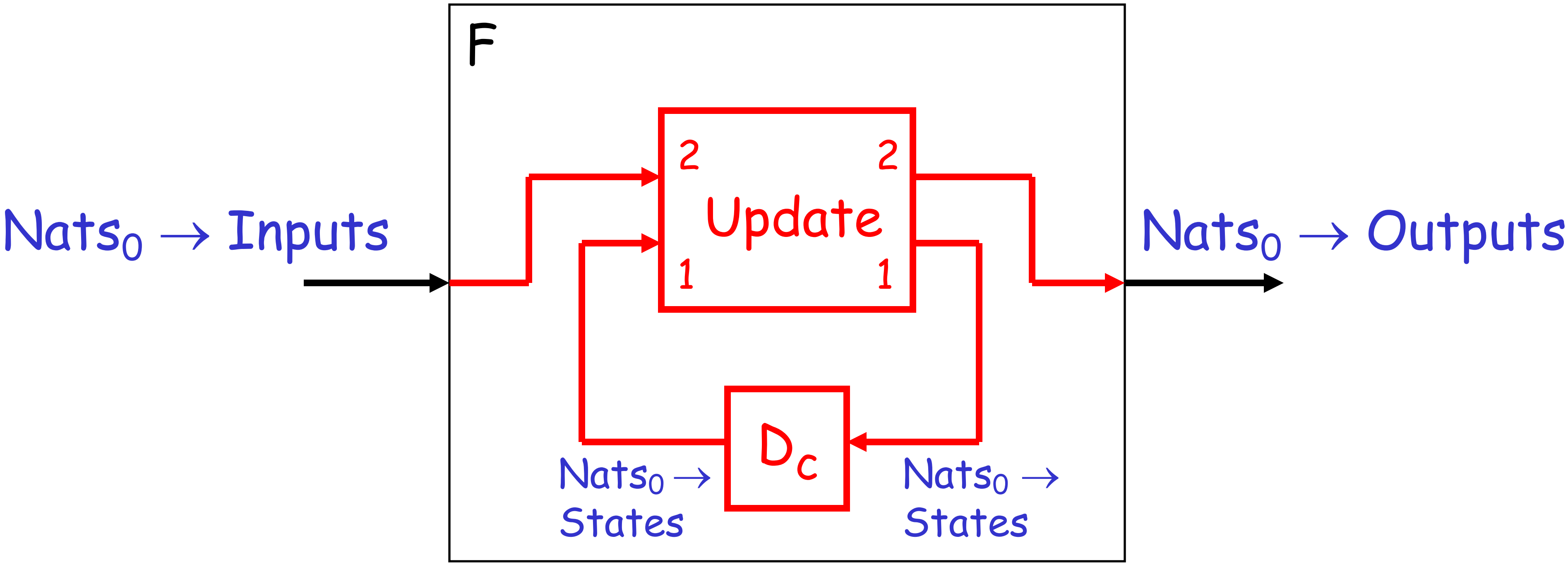
such as continuous-time delay,

are not naturally viewed naturally as transition systems.)

A Discrete-Time Reactive System

$$F : [\, Nats_0 \rightarrow Inputs \,] \rightarrow [\, Nats_0 \rightarrow Outputs \,]$$

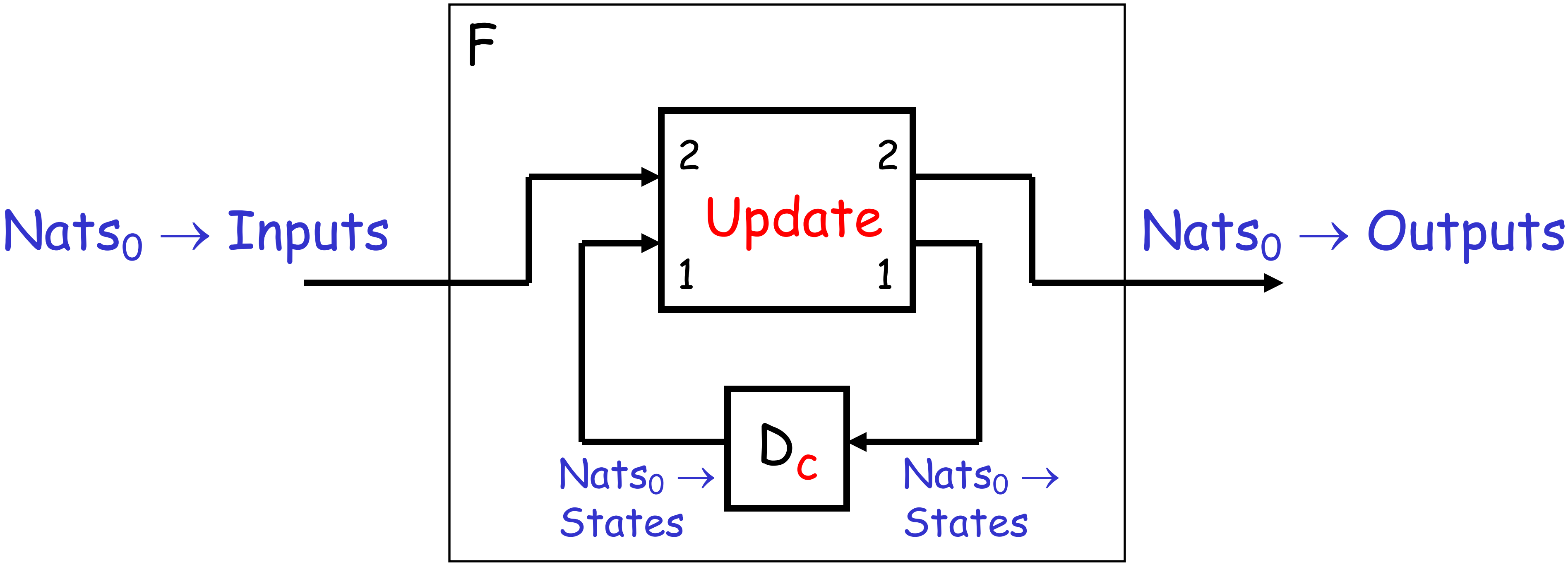
State-Based Implementation



Update: memory-free

Memory = States

State Implementation



update : States $\times$ Inputs $\to$ States $\times$ Outputs

c $\in$ States ("initial state")

State Implementation of the Parity System

Inputs = Bools

Outputs = Bools

States = Bools

initialState = true

update : $States \times Inputs \to States \times Outputs$

such that $\forall\, q \in States,\ \forall\, x \in Inputs,$

$$update\,(q,x)_1 = (\, q \neq x \,)$$

$$update\,(q,x)_2 = q$$

State Implementation of the Count System

Inputs = Bools

Outputs = Bools

States = Ints

initialState = 0

update : States $\times$ Inputs $\rightarrow$ States $\times$ Outputs

such that $\forall\, q \in$ States, $\forall\, x \in$ Inputs,

$$\text{update}(q,x)_1 = \pm(x,q)$$

$$\text{update}(q,x)_2 = \text{pos}(q)$$

A State Machine

Inputs (set of possible input values)

Outputs (set of possible output values)

States (set of states)

initialState $\in$ States

update : States $\times$ Inputs $\rightarrow$ States $\times$ Outputs

A state machine is

finite

iff

States is a finite set.

Parity : can be implemented by a two-state machine.

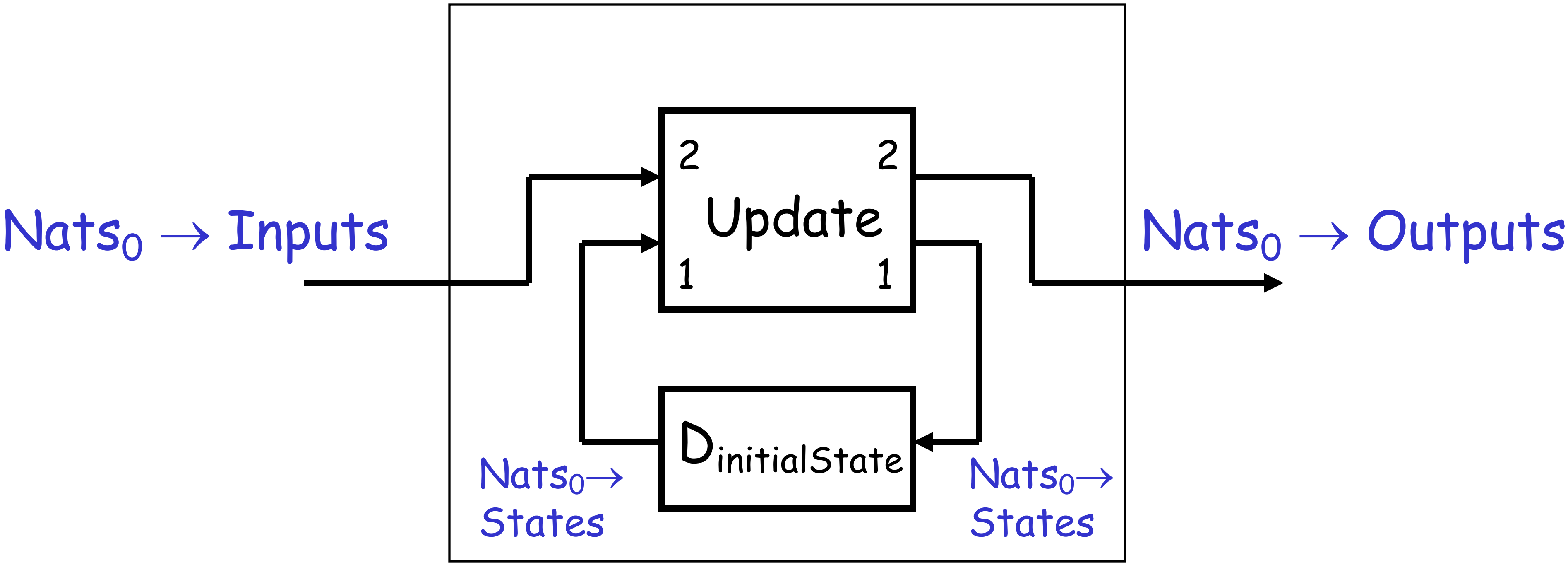
Count : cannot be implemented by finite-state machine.

Discrete-Time Reactive Systems

Every memory-free system can be implemented by a one-state machine.

Every causal system can be implemented by a state machine (take as state the entire history of inputs), and every system that can be implemented by a state machine is causal.

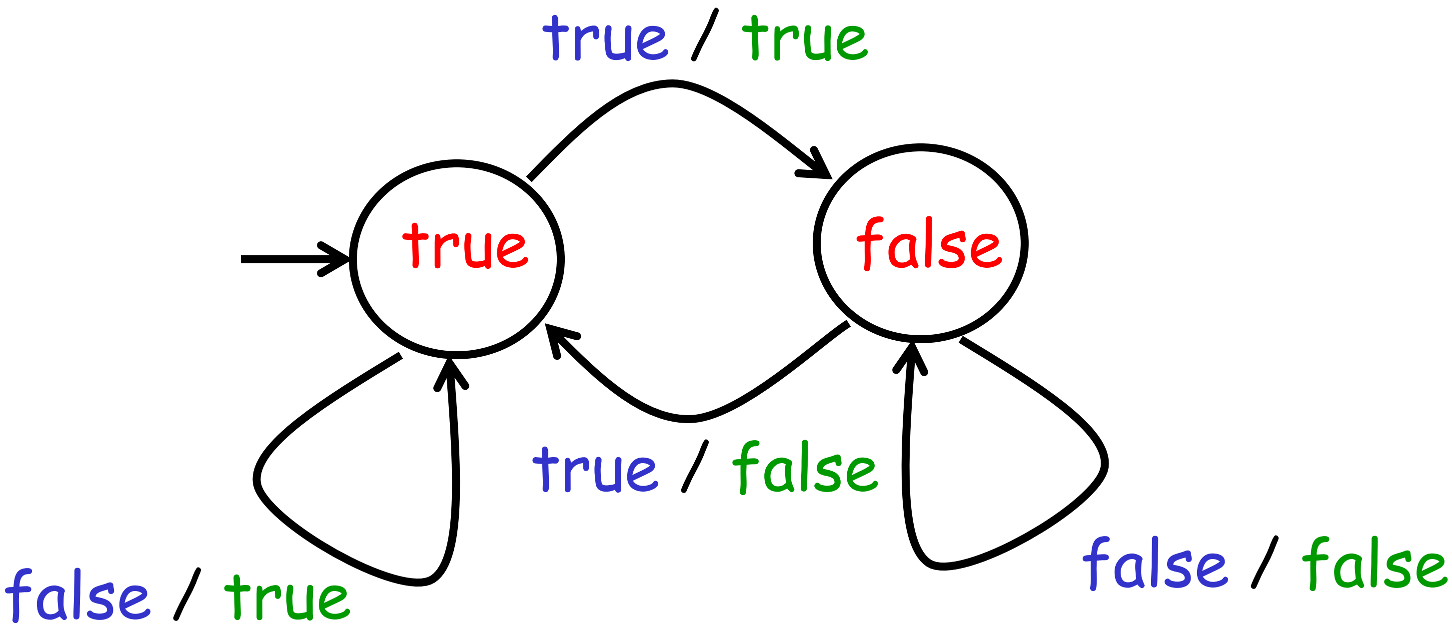
The Block Diagram of a State Machine



The Transition Table of a Finite-State Machine (Parity)

Current state	Input	Next state	Output
true	true	false	true
true	false	true	true
false	true	true	false
false	false	false	false

The Transition Diagram of a State Machine (Parity)



States = Bools
Inputs = Bools
Outputs = Bools