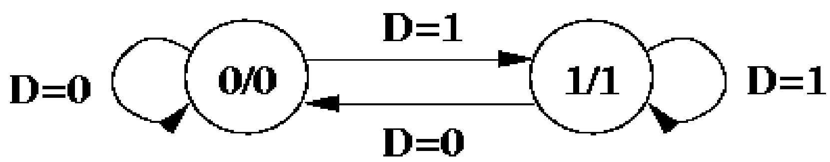
Finite state machines: flip-flop

Flip-flop can be modeled as a finite-state machine



D flip-flop

State: 1 bit (q)
Input: 1 bit (D)
Output: current state (z)

q	D	q^+	z	T
0	0	0	0	0
0	1	1	0	1
1	0	0	1	1
1	1	1	1	0

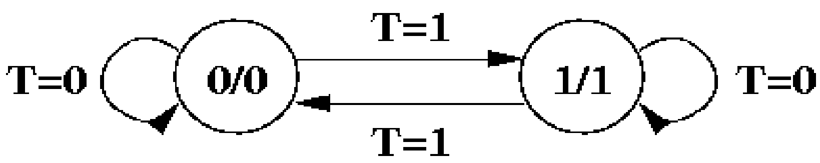
In state 0:
- **input 0 gives new state 0 (reset)**
- **input 1 gives new state 1 (set)**

In state 1:
- **input 0 gives new state 0 (reset)**
- **input 1 gives new state 1 (set)**

Could implement this in the obvious way with a D flip-flop, or use a T flip-flop!

Finite state machines: flip-flop

T flip-flop:



State: 1 bit (q)
Input: 1 bit (T)
Output: current state (z)

q	T	q^+	z	D
0	0	0	0	0
0	1	1	0	1
1	0	1	1	1
1	1	0	1	0

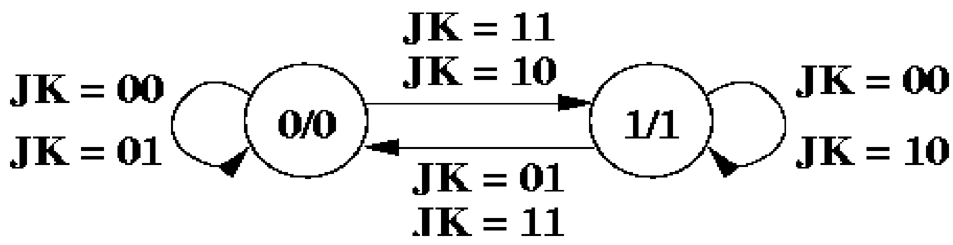
In state 0:
input 0 gives new state 0 (hold)
input 1 gives new state 1 (toggle)

In state 1:
input 0 gives new state 1 (hold)
input 1 gives new state 0 (toggle)

Note that we can implement either type of flip-flop with the other type
and some combinational circuit

Finite state machines: flip-flop

JK flip-flop:



State: 1 bit (q)
Input: 2 bits (JK)
Output: current state (z)

q	J	K	q^+	z	
0	0	0	0	0	hold
0	0	1	0	0	reset
0	1	0	1	0	set
0	1	1	1	0	toggle
1	0	0	1	1	hold
1	0	1	0	1	reset
1	1	0	1	1	set
1	1	1	0	1	toggle

Input is 2 bits, so there are 4 outgoing arcs from each state, but 2 pairs are equivalent
This can be implemented with either a T or D flip-flop