Finite State Systems

In this lecture we take our first look at finite state systems.

In the process we will discuss digital image processing and see how a finite state machine can specify a basic image filter.

Reactive Systems

- Wait to receive an input
- Respond with:
  - an output (changing to a new state) or
  - change to new state without output
- Response depends on (finite) history

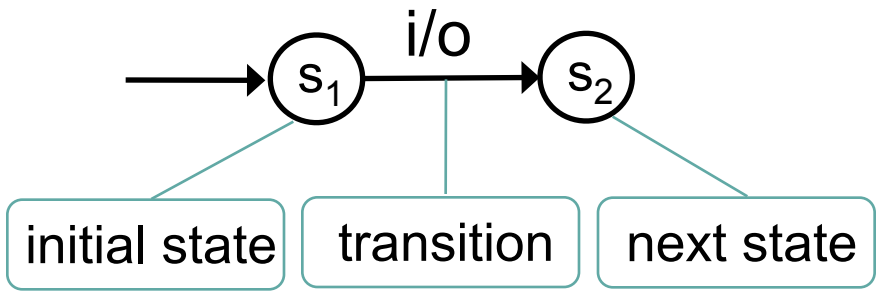
Finite State Machines

- A conceptual tool for modelling reactive systems.
- Not limited to software systems.
- Pre-dates computing science.
- Used to specify required system behaviour in a precise way.
- Then implement as software/hardware (and perhaps verify against FSM).

Formal Definition

FSM transducer model, M, consists of:

- Set of input symbols, Σ ("input alphabet")
- Set of output symbols, Λ ("output alphabet")
- Set of states, Q (one identified as initial)
- Set of transitions, T, each of the form $(s_{i-1}, i_i/o_i, s_i)$ (ε used when output is empty)



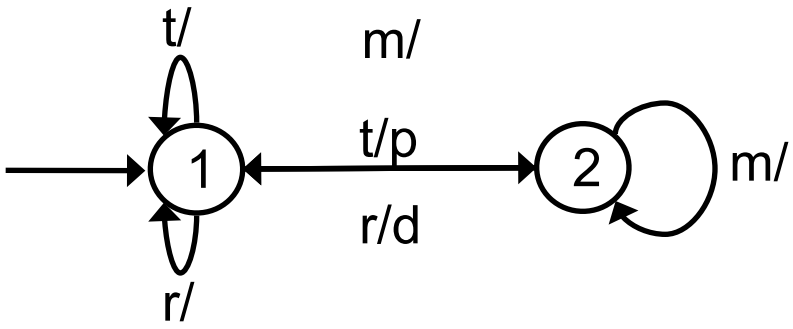
Parking Meter Example

Σ = {m,t,r} money, ticket request, refund request

Λ = {p,d} print ticket, deliver refund

Q = {1,2}

T = {(1,t/ε,1), (1,r/ε,1), (1,m/ε,2), (2,t/p,1), (2,r/d,1), (2,m/ε,2)}

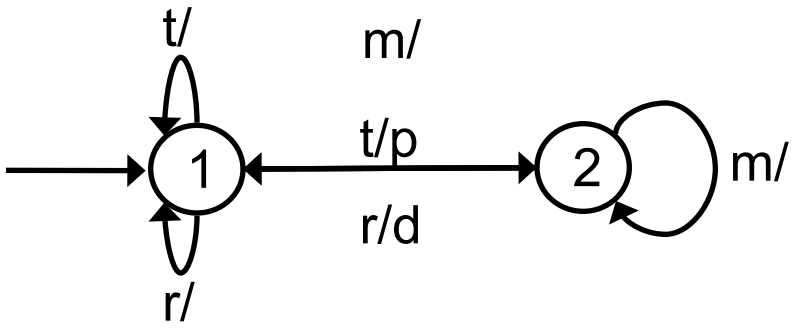


This is a **transducer** FSM because it has some outputs.

FSM Traces

- Finite sequence of alternating state and transition labels, starting and ending with a state: $[s_0, i_1/o_1, s_1, i_2/o_2, s_2, \ldots s_{n-1}, i_n/o_n, s_n]$
- s_0 is the initial state.
- Each $[s_{i-1}, i_i/o_i, s_i]$ subsequence must appear as a transition in T

Parking Meter Trace Example



Traces include:

[1, m/, 2, t/p, 1]
[1, m/, 2, m/, 2, r/d, 1]
[1, m/, 2, t/p, 1, m/, 2, m/, 2]
[1, t/, 1, t/, 1, m/, 2]
… etc

Behaviour of FSM is the set of all possible traces.
This is not necessarily a finite set.

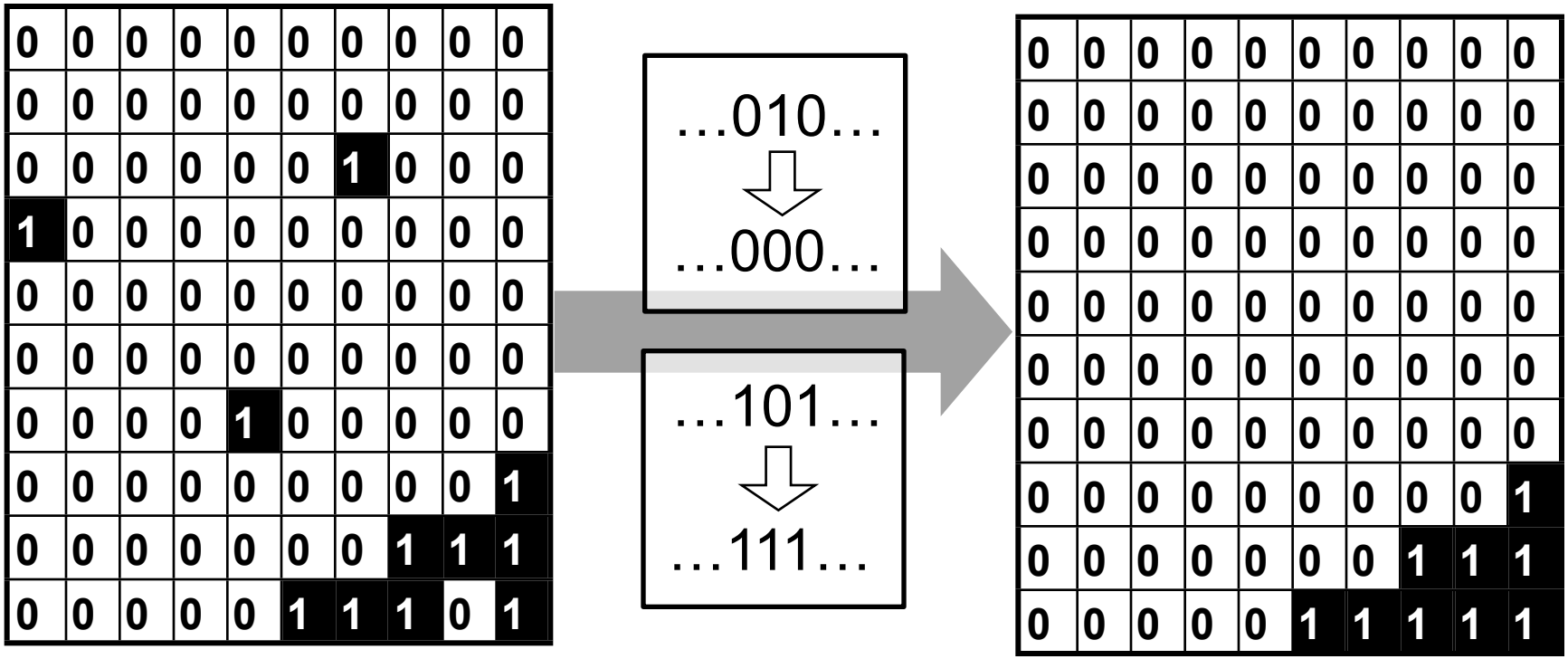
Smoothing a Digital Image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1	0	1

Noise

Object

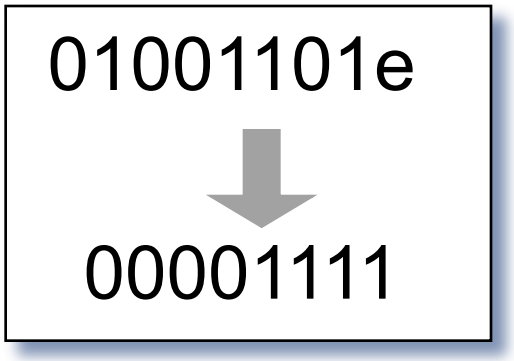
Simple Filter



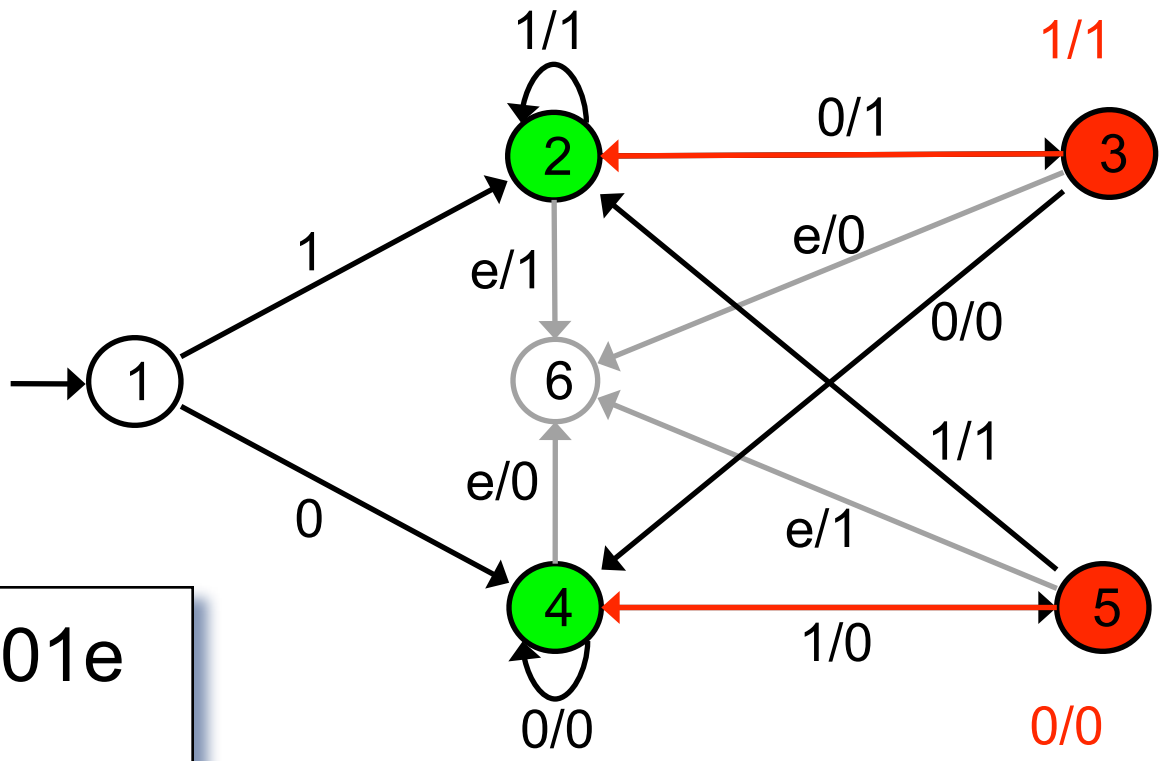
Simple Filter Specification

Assume that the character “e” marks the end of the input sequence. Also assume that the first character of the sequence is not noise. Ensure that any isolated “1” or “0” in the input sequence is inverted. All other characters in the input sequence translate directly to the output.

Example:



FSM For Filter



01001101e

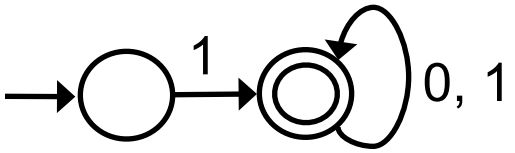
↓

00001111

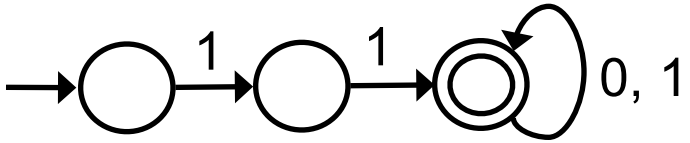
Limits of FSMs

- No memory on states or transitions
- so "memory" is simply the location in the transition network.
- Consequently, FSMs can't do things like "count" over arbitrarily large sequences.

All strings beginning with one 1



All strings beginning with two 1's



All strings with equal numbers of 0's and 1's? No FSM possible