Finite-State Machines (Automata) lecture 12

- a simple form of computation
- used widely
- one way to find patterns
- one way to describe reactive systems

Reactive Systems

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

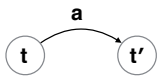
Finite State Machines

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

Formal Definition DFA

Deterministic FSM, deterministic finite automaton
$D = (Q, s_0, F, \Sigma, N)$

- Set of states, Q
- Initial state $s_0 \in Q$
- Accepting states $F \subseteq Q$
- Alphabet Σ
- Next state function, $t' = N(t, a)$
  where $t, t' \in Q$
  and $a \in \Sigma$.

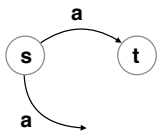


A formal mathematical definition of a finite state machine

Formal Definition NFA

Non-deterministic FSM,
non-deterministic finite automaton
$N = (Q, s_0, F, \Sigma, T)$

- Set of states, Q
- Initial state $s_0 \in Q$
- Accepting states $F \subseteq Q$
- Alphabet Σ
- Transition relation, T(s, a, t)
  where s, t $\in Q$
  and $a \in \Sigma \cup \{\varepsilon\}$.



A formal mathematical definition of a finite state machine

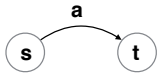
Formal Definition

Every DFA is a NFA

Deterministic FSM,
$D = (Q, s_0, F, \Sigma, N)$

Non-deterministic FSM,
$N = (Q, s_0, F, \Sigma, T)$

- Set of states, Q
- Initial state $s_0 \in Q$
- Accepting states $F \subseteq Q$
- Alphabet Σ
- T(s, a, t) iff t = N(s, a)



A formal mathematical definition of a finite state machine

Deterministic FSMs

Many authors give an informal definition of deterministic

- all states have no more than one transition leaving the state for each input symbol.

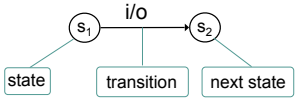
Formal definition says, exactly one state …

- We consider the informal presentation to include an implicit “black hole”, or “sink” state, from which there is no escape.
- Where there is no explicit transition for a symbol, it takes us to the black hole.

Formal Definition

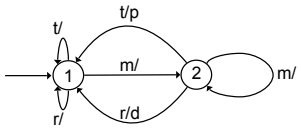
FSM transducer model, M, consists of:

- Set of states, Q
- Initial state $s_0 \in Q$
- Alphabets of input and output symbols i/o
- Transition relation, T(s, a, t) where s, t $\in Q$ and a $\in$ (In $\cup \{\varepsilon\}$) x (Out $\cup \{\varepsilon\}$).



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)}

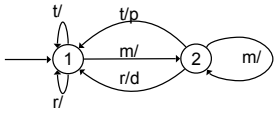


This is a **transducer** FSM because it has some outputs.

FSM Traces

- Finite sequence of state and transition labels, starting, alternating, 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



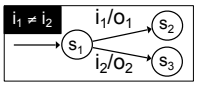
Traces include:

[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*

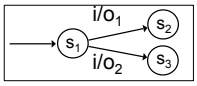
Behaviour of FSM is the set of all possible traces.
This is not necessarily a finite set.

Determinism

- In a deterministic FSM, all states have no more than one transition leaving the state for each input symbol.



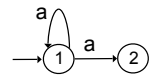
- In a non-deterministic FSM, states may have more than one transition leaving to different successor states for the same input symbol.
- Sometimes non-deterministic FSMs are easier to define.
- Can always convert from a non-deterministic to a deterministic FSM.



Determinism and Traces

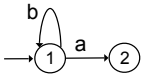
A FSM, M, is deterministic if for every string $x \in \Sigma^*$
there is at most one trace for x in M
(where Σ^* is the set of all strings in alphabet of M)

non-deterministic (choice)



Sequence	Trace
aa	[1,a,1,a,2]
aa	[1,a,1,a,1]

deterministic (no choice)



Sequence	Trace
ba	[1,b,1,a,2]
bb	[1,b,1,b,1]

Acceptors

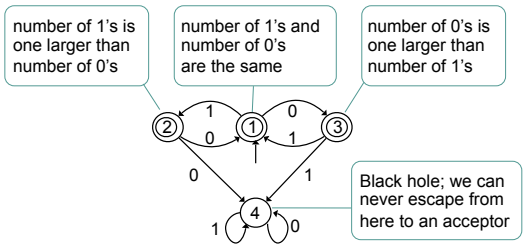
Definition as before but:

- Empty output alphabet (all outputs are ε)
- Some states marked as accepting.

Input sequence is accepted if there is a trace from the initial state to an acceptor state.

Language of the FSM is the set of sequences it accepts.

Acceptor Example

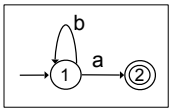


Accepts strings of 0's and 1's for which the difference between number of 0's and number of 1's in a subsequence is at most 1.

Language of a Deterministic FSM Acceptor

Set of strings whose (unique) traces end in an accepting state ($s_a \in F$)

Accepts: ba
bba
bbba
…*etc*



Rejects: aba (no trace)
bbb (trace but not ending in accepting state)