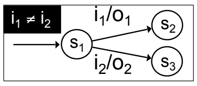
Languages and Automata

cl

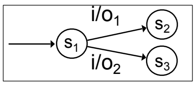
- deterministic machines
- languages and machines
- the Boolean algebra of languages
- non-determinism

Determinism

DFA



NFA

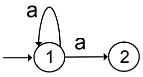


- In a deterministic machine (DFA), all states have exactly one transition leaving the state for each input symbol.
- In a non-deterministic machine (NFA), each state may have any number of transitions leaving to different successor states for the same input symbol.
- Sometimes NFA are easier to define.
- Can always convert from a NFA to a deterministic DFA.

Determinism and Traces

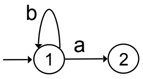
A FSM, M, is deterministic if for every string $x \in \Sigma^*$ there is exactly one trace for x in M
(where Σ^* is the set of all strings in alphabet of M)

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]

Determinism

If we have a machine with at most one transition for each (s,a) pair, we can always convert to an equivalent DFA for which every state has exactly one transition leaving the state for each input symbol.

For this machine there is exactly one trace for each input string

- Proof

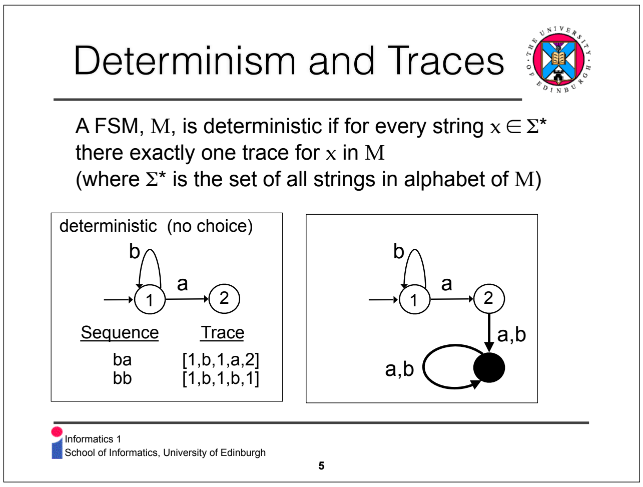
Add a new “black hole” state, ●

For every pair (s, a) for which there is no state t with a transition T(s, a, t), add a transition T(s, a, ●).

This includes a transition T(●, a, ●) for each a ∈ Σ . You cannot escape from the black hole.

The black hole ● is not an accepting state.

This machine accepts the same language as the original.



Determinism and Traces

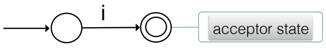
A FSM, M, is deterministic if for every string $x \in \Sigma^*$ there exactly one trace for x in M (where Σ^* is the set of all strings in alphabet of M)

deterministic (no choice)

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

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.

The language accepted by a machine



acceptor state

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.

$$L(M) \subseteq \Sigma^*$$

Two machines are **equivalent** if they define the same language.

Regular languages

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

$$L(M) \subseteq \Sigma^*$$

We say $A \subseteq \Sigma^*$ is **regular** iff

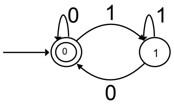
$$A = L(M)$$

for some machine M.

A language is regular iff it is the language accepted by some DFA

Two examples

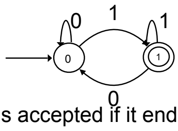
	×2	×2 + 1
	0	1
0	0	1
1	0	1



Even
binary
numbers

Input sequence is accepted if it ends with a zero.

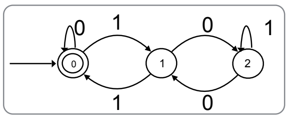
	×2	×2 + 1
	0	1
0	0	1
1	0	1



Odd
binary
numbers

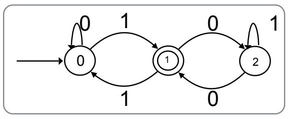
Input sequence is accepted if it ends with a one.

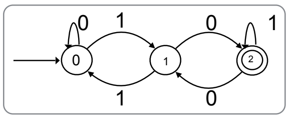
Three examples



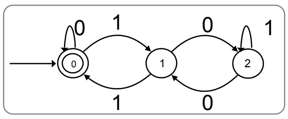
Which binary numbers are accepted?

	×2	×2 + 1
mod 3	0	1
0	0	1
1	2	0
2	1	2

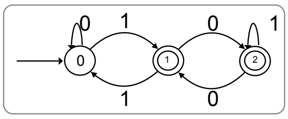




By three or not by three?

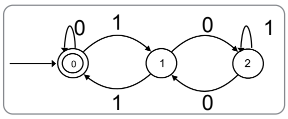


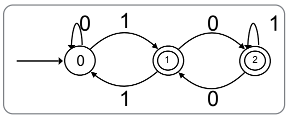
divisible by three



not
divisible by three

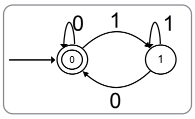
The complement of a regular language is regular

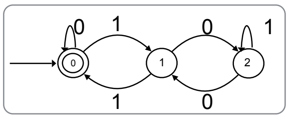




If $A \subseteq \Sigma^*$ is recognised by M then $\overline{A} = \Sigma^* \setminus A$ is recognised by $\overline{\mathbf{M}}$ where $\overline{\mathbf{M}}$ and **M** are identical except that the accepting states of $\overline{\mathbf{M}}$ are the non-accepting states of **M** and vice-versa

The intersection of two regular languages is regular





divisible by 6

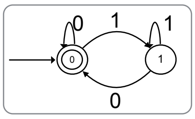
≡

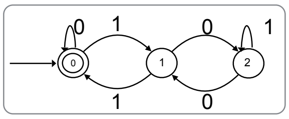
divisible by 2

and

divisible by 3

The intersection of two regular languages is regular



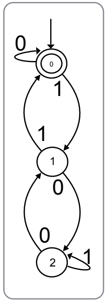


Run both machines in parallel?

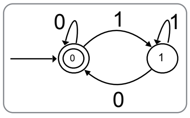
Build one machine that simulates two machines running in parallel!

Keep track of the state of each machine.

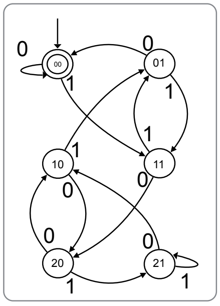
The intersection of two regular languages is regular



||

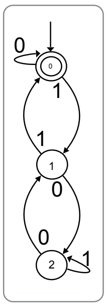


=

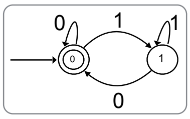


mod 3	mod 2	0		1	
		3	2	3	2
0	0	0	0	1	1
1	0	2	0	0	1
2	0	1	0	2	1
0	1	0	0	1	1
1	1	2	0	0	1
2	1	1	0	2	1

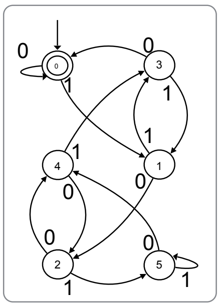
The intersection of two regular languages is regular



||

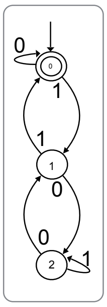


=

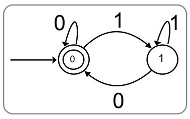


mod 3	mod 2	mod 6	0			1		
			3	2	6	3	2	6
0	0	0	0	0	0	1	1	1
1	0	4	2	0	2	0	1	3
2	0	2	1	0	4	2	1	5
0	1	3	0	0	0	1	1	1
1	1	1	2	0	2	0	1	3
2	1	5	1	0	4	2	1	5

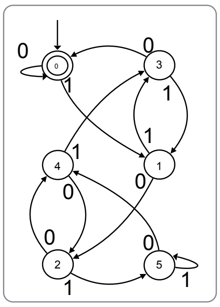
The intersection of two regular languages is regular



||



=



mod 3	mod 2	mod 6	0			1		
			3	2	6	3	2	6
0	0	0	0	0	0	1	1	1
1	1	1	2	0	2	0	1	3
2	0	2	1	0	4	2	1	5
0	1	3	0	0	0	1	1	1
1	0	4	2	0	2	0	1	3
2	1	5	1	0	4	2	1	5

The regular languages $A \subseteq \Sigma^*$ form a Boolean Algebra

- Since they are closed under intersection and complement.

Determinism

Can always convert a machine with at most one transition for each (s,a) pair to an equivalent DFA for which every state has exactly one transition leaving the state for each input symbol.

For this machine there is exactly one trace for each input string

- Proof Add a new “black hole” state, ●

  For every pair (s, a) for which there is no state t with a transition T(s, a, t), add a transition T(s, a, ●).

  This includes a transition T(●, a, ●) for each a ∊ Σ . You cannot escape from the black hole.

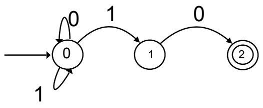
  The black hole ● is not an accepting state.

This machine accepts the same language as the original.

We can simulate each trace. If we are in the black hole state in the new machine then we are in no state in the old machine.

Non Determinism

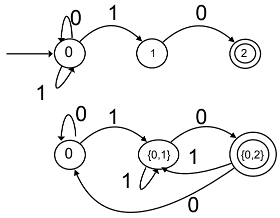
In a non-deterministic machine (NFA), each state may have any number of transitions with the same input symbol, leaving to different successor states.



	0	1
0	0	0,1
1	2	
2		

Non Determinism

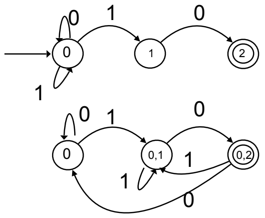
In a non-deterministic machine (NFA), each state may have any number of transitions with the same input symbol, leaving to different successor states.



	0	1
0	0	0,1
1	2	
2		
0,1	0,2	0,1
0,2	0	0,1

Non Determinism

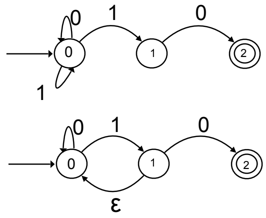
We can simulate a non-deterministic machine using a deterministic machine – by keeping track of the set of states the NFA could possibly be in.



	0	1
0	0	0,1
1	2	
2		
0,1	0,2	0,1
0,2	0	0,1

Internal Transitions

We sometimes add an internal transition ε to a non-deterministic machine (NFA)This is a state change that consumes no input.



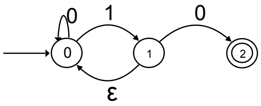
	0	1	ε
0	0	1	
1	2		0
2			

Internal Transitions

We sometimes add **internal transitions** – labelled ε – to a non-deterministic machine (NFA).

This is a state change that consumes no input.

It introduces non-determinism in the observed behaviour of the machine.

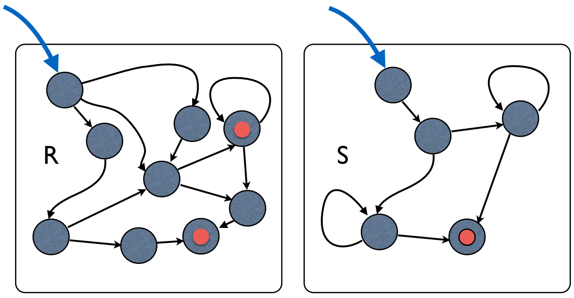


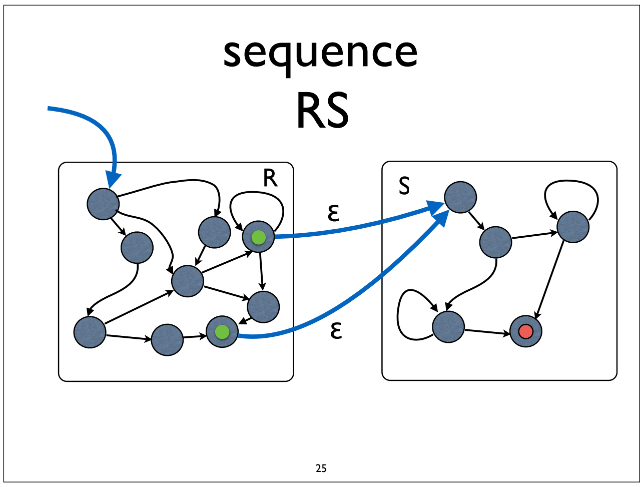
	0	1	ε
0	0	1	
1	2		0
2			

	0ε*	1ε*
0	0	1,0
1	2	
2		

NFA single start state

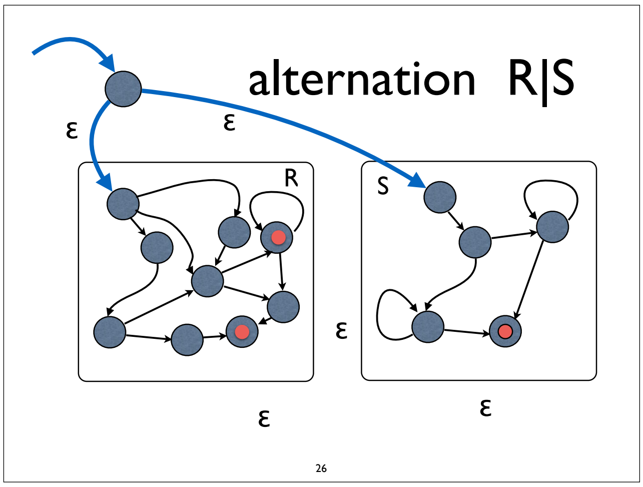
any number of accepting states



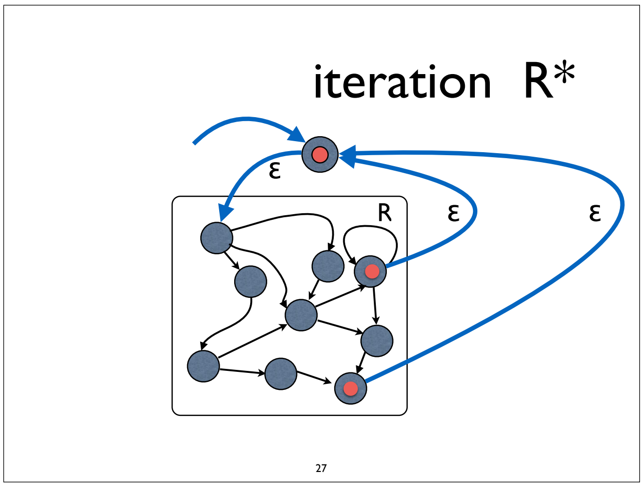


sequence
RS

The red lines are automatic transitions that can always happen, without any input. They are normally labelled ε



The red lines are automatic transitions that can always happen, without any input. They are normally labelled ε



iteration R*

The red lines are automatic transitions that can always happen, without any input. They are normally labelled ε