NFA to DFA

Cl

- the subset construction
- ε-transitions

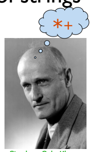
---

regular expressions

each regex is a pattern that matches a set of strings

- any character is a regex
  - matches itself
- if `R` and `S` are regex, so is `RS`
  - matches a match for `R` followed by a match for `S`
- if `R` and `S` are regex, so is `R|S`
  - matches any match for `R` or `S` (or both)
- if `R` is a regex, so is `R*`
  - matches any sequence of 0 or more matches for `R`

Kleene *



Stephen Cole Kleene
1909-1994

- The algebra of regular expressions also includes elements `0` and `1`
  - `0 = Ø` matches nothing; `1 = Σ*` matches everything
  - `ε = Ø*` matches the empty string

```
0|R = R|0 = R    1|R = R|1 = 1      (S|T)R = SR|TR
 0R = R0 = 0      εR = Rε = R        R(S|T)= RS|RT
R|S = S|R          ε = 0*      A* = ε|AA* = ε|A*A
```

the language of strings that match a regex, R, is recognised by some ε-FSM

A mathematical definition of a
Finite State Machine.
$M = (Q, \Sigma, \Delta, S, F)$

Q: the set of states,
Σ: the alphabet of the machine
- the tokens the machine can process,

Δ: the set of **transitions**
is a set of (state, symbol, state) triples
$\Delta \subseteq Q \times \Sigma \times Q$.
S: the set of beginning or **start** states of the machine
F: the set of the machine's accepting of **finish** states.

A *trace* for $s = \langle x_0, \ldots x_{k-1} \rangle \in \Sigma^*$ (a string of length k)
is a sequence of k+1 states $\langle q_0, \ldots q_k \rangle$
such that $(q_i, x_i, q_{i+1}) \in \Delta$ for each $i < k$

$M = (Q, \Sigma, \Delta, B, A,)$

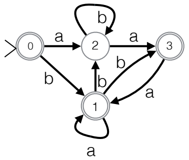
A *trace* for $s = \langle x_0, \ldots, x_{k-1} \rangle \in \Sigma^*$ (a string of length k)
is a sequence of $k+1$ states $\langle q_0, \ldots q_k \rangle$
such that $(q_i, x_i, q_{i+1}) \in \Delta$ for each $i < k$

We say s is *accepted* by M
iff there is
a trace $\langle q_0, \ldots q_k \rangle$ for s
such that $q_0 \in B$ and $q_k \in A$

Definition FSM

non-deterministic finite state automaton FSM

states — a set of states
sigma — a set of symbols
delta ⊆ (states × sigma × states)
start ⊆ states — starting states
accept ⊆ states — accepting states

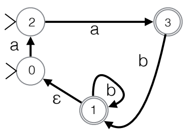



```
FSM qs as ts es ss fs where
  qs = [0..3]
  as = "ab"
  ts = [(0,'b',1),(0,'a',2),(1,'a',1),(1,'b',2)
       ,(1,'b',3),(2,'a',3),(2,'b',2),(3,'a',1)]
  ss = [0]
  fs = [0,1,3]
```

Definition ε-FSM

finite state automaton FSM
with ε-transitions

qs states — a set of states
as sigma — a set of symbols
ts delta ⊆ (states × sigma × states)
es epsilon ⊆ (states × states)
ss start ⊆ states — starting states
fs final ⊆ states — accepting states




```
EPS qs as ts es ss fs where
  qs = [0..3]
  as = "ab"
  ts = [(0,'a',2), (1,'b',1), (2,'a',3),(3,'b',1)]
  es = [(1,0)]
  ss = [0,2]
  fs = [1,3]
```

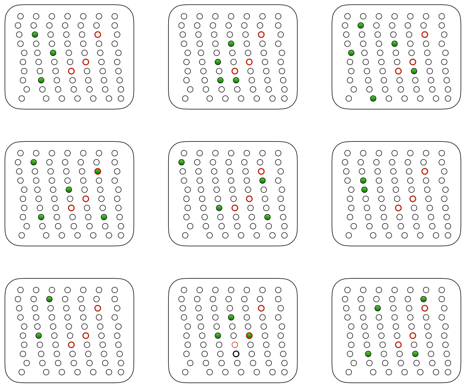
Definition DFA

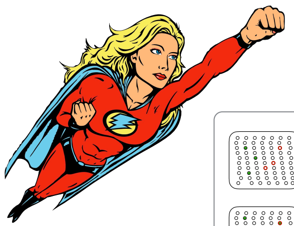
is a finite state automaton
(FSA, without ε)

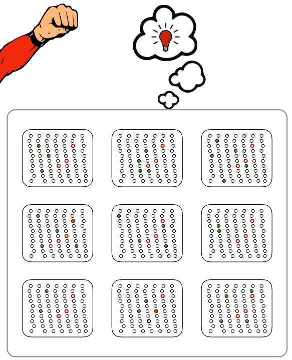
states — a set of states
sigma — a set of symbols
delta ⊆ (states × sigma × states)
start ⊆ states — starting states
accept ⊆ states — accepting states

A deterministic machine has
- no ε-transitions
- exactly one starting state
- for each (state, symbol) pair, (q, s)
  exactly one transition of the form (q, s, q')

a DFA can be efficiently implemented
in software or hardware

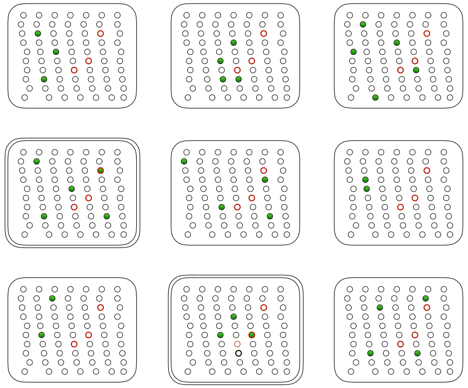






superstates

a **superstate** is a set of states



superstates

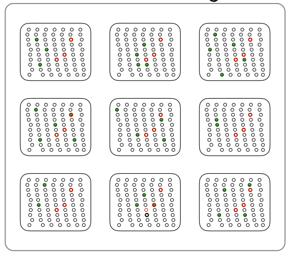
a **superstate** is a set of states

superstates are the states of DFA

The set of start states is the unique start superstate

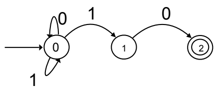
A finish superstate is any superstate that includes a finish state

A transition is the move from one set of lit lights to the next

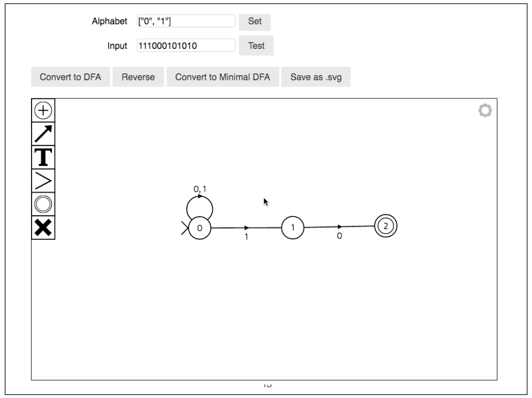


Non Determinism

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

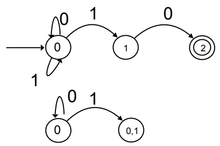


	0	1
0	0	0,1
1	2	
2		



Non Determinism

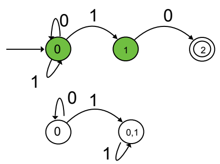
In a non-deterministic machine (NFA), each state may have any number of transitions with the same input symbol, leaving to different successor states.



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

Non Determinism

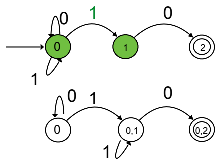
In a non-deterministic machine (NFA), each state may have any number of transitions with the same input symbol, leaving to different successor states.



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

Non Determinism

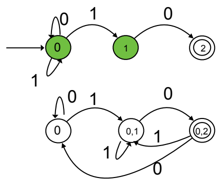
In a non-deterministic machine (NFA), each state may have any number of transitions with the same input symbol, leaving to different successor states.



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

Non Determinism

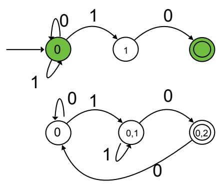
In a non-deterministic machine (NFA), each state may have any number of transitions with the same input symbol, leaving to different successor states.



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

Non Determinism

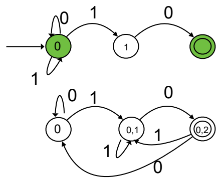
We can simulate a non-deterministic machine using a deterministic machine – by keeping track of the set of states the NFA could possibly be in.



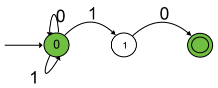
	0	1
0	0	0,1
1	2	
2		
0,1	0,2	0,1
0,2	0	

Non Determinism

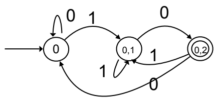
We can simulate a non-deterministic machine using a deterministic machine – by keeping track of the set of states the NFA could possibly be in.



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

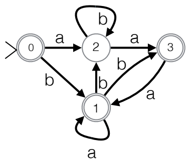


```
FSM qs as ts es ss fs where
  qs = [0..2]
  as = "01"
  ts = [(0,'0',0),(0,'1',0)
       ,(0,'1',1),(1,'0',2)]
  ss = [0]
  fs = [2]
```

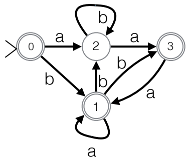


```
FSM qs as ts es ss fs where
  qs = [[0],[0,1],[0,2]]
  as = "01"
  ts = [([0],'0',[0])
       ,([0],'1',[0,1])
       ,([0,1],'0',[0,2])
       ,([0,1],'1',[0,1])
       ,([0,2],'0',[0])
       ,([0,2],'1',[0,1])]
  ss = [[0]]
  fs = [[0,2]]
```

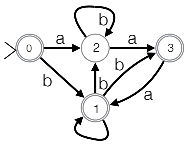
	a	b
0	2	1
1	1	2,3
2	3	2
3	1	
2,3		

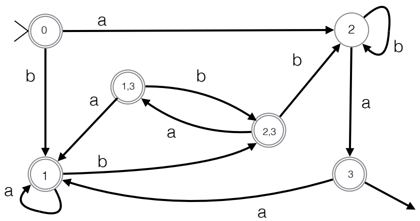


	a	b
0	2	1
1	1	2,3
2	3	2
3	1	
2,3	1,3	2



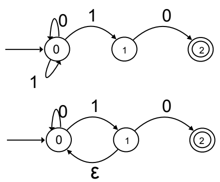
	a	b
0	2	1
1	1	2,3
2	3	2
3	1	
2,3	1,3	2
1,3	1	2,3





Internal Transitions

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



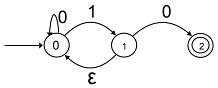
	0	1	ε
0	0	1	
1	2		0
2			

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			

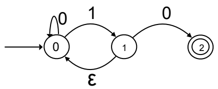
	0ε*	1ε*
0	0	1,0
1	2	
2		

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.

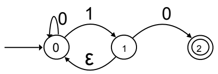


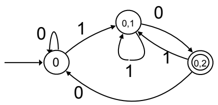
	0	1	ε
0	0	1	
1	2		0
2			

	0ε*	1ε*
0	0	0,1
0,1	0,2	0,1
0,2	0	0,1

Internal Transitions

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

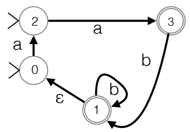




	0	1	ε
0	0	1	
1	2		0
2			

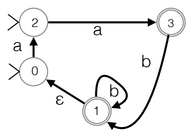
	0ε*	1ε*
0	0	0,1
0,1	0,2	0,1
0,2	0	0,1

	a	b	ε
0	2		
1		1	0
2	3		
3		1	

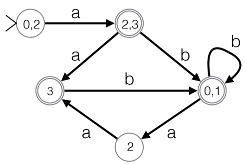


	aε*	bε*
0,2	2,3	
2,3		

	a	b	ε
0	2		
1		1	0
2	3		
3		1	



	aε*	bε*
0,2	2,3	
2,3	3	0,1
3		0,1
0,1	2	0,1
2	3	



http://xkcd.com/