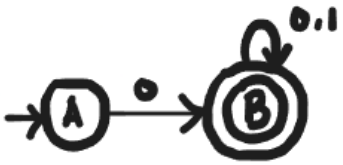
NFA → DFA

Every DFA is an NFA, but not vice versa

L = { Set of all strings over (0,1) that starts with '0' }



Step 1: Write down the NFA state transition table

	0	1
→A	B	ϕ
*B	B	B

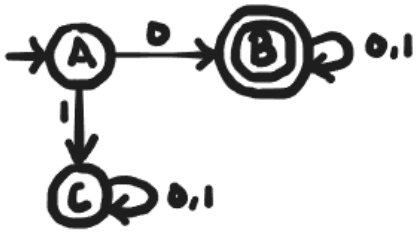
Step 2: Convert it to a DFA table (NFA table serves as a basis)

	0	1
→A	B	C
*B	B	B
C	C	C

NOTE:

ϕ becomes a new state

Step 3: Draw the new DFA diagram



NFA → DFA

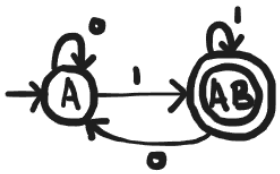
L = { Set of all strings over (0,1) that ends with '1' }

	0	1
→A	A	A,B
*B	-	-

→

	0	1
→A	A	AB
*AB	A	AB

NOTE:
- We combine 2 diff. states (union) if they point to the same state in the NFA
- Build upon the current DFA table but still use the NFA table as basis

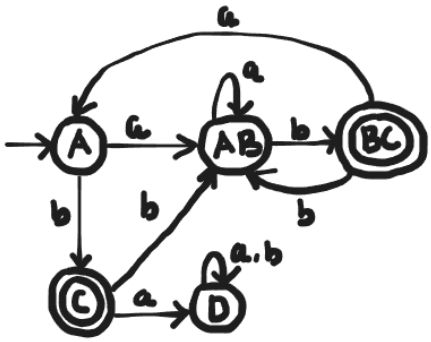


Find the equivalent DFA for the NFA given by M= [{A,B,C}, (a,b), δ, A, {C}] where δ is given by:

	a	b
→A	A,B	C
B	A	B
©C	-	A,B

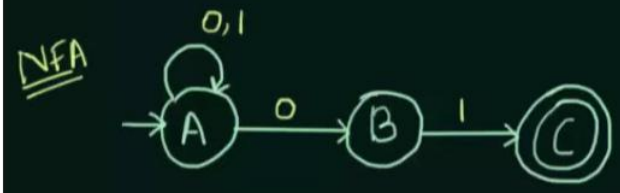
	a	b
→A	AB	C
AB	AB	BC
*BC	A	AB
*C	D	AB
D	D	D

NOTE:
If ever the final state gets combined with another state, that state also becomes a final state



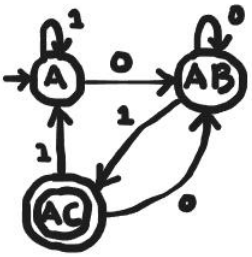
NFA → DFA

L = { Set of all strings over (0,1) that ends with '01' }. Construct its equivalent DFA

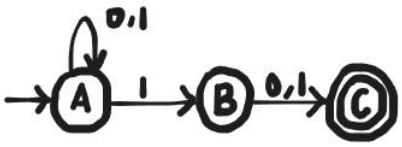


	0	1
→A	A, B	A
B	ϕ	C
C	ϕ	ϕ

	0	1
→A	AB	A
AB	AB	AC
* AC	AB	A

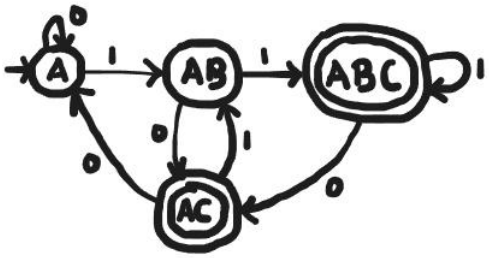


Design an NFA for a language that accepts all strings over {0,1} in which the second last symbol is always '1'. Then convert it to its equivalent DFA.



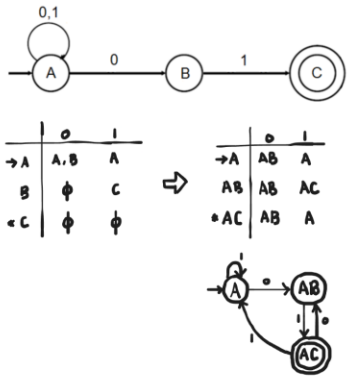
	0	1
→ A	A	A,B
B	C	C
* C	ϕ	ϕ

	0	1
→ A	A	AB
AB	AC	ABC
* AC	A	AB
* ABC	AC	ABC



EXAMPLE 1

- Convert the given NFA into DFA



	0	1
→A	A, B	A
B	ϕ	C
*C	ϕ	ϕ

⇨

	0	1
→A	AB	A
AB	AB	AC
*AC	AB	A

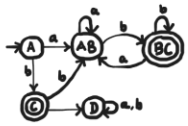
EXAMPLE 2

- Find the equivalent DFA for the NFA given by M = [{A,B,C}, (a,b), δ, A, {C}] where δ is given by:

	a	b
→ A	A,B	C
B	A	B
* C	-	A,B

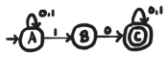
⇨

	a	b
→A	AB	C
AB	AB	BC
*BC	AB	BC
*C	D	AB
D	D	D



EXAMPLE 3

- Design an NFA for a language and convert it to its equivalent DFA
  - L = set of all strings over {0,1} that contains 10



	0	1
→A	A	A,B
B	C	ϕ
C	C	C

⇨

	0	1
→A	A	AB
AB	AC	AB
*AC	AC	ABC
*ABC	AC	ABC