Step-by-step Conversion of Regular Expressions to C Code

On the regular expression:

$$((a \cdot b)|c)^*$$

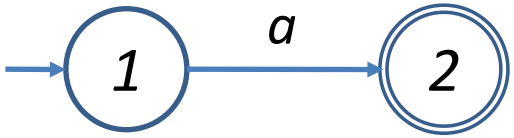
THOMPSON’S CONSTRUCTION

Convert the regular expression to an NFA.

Step 1: construct NFA for r_1.

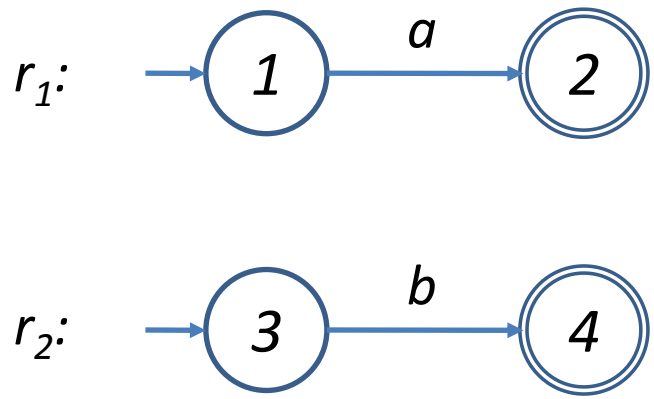
$$(\ (\underbrace{a}_{r_1} \cdot b) \ | \ c \ )^*$$

r_1:



Step 2: construct NFA for r_2.

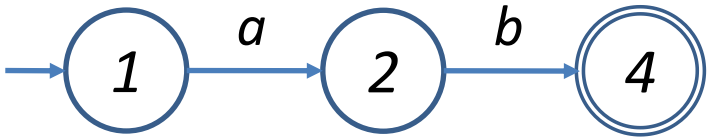
$$(\; (\underbrace{a}_{r_1} \cdot \underbrace{b}_{r_2}) \; | \; c \;)^*$$



Step 3: construct NFA for r_3.

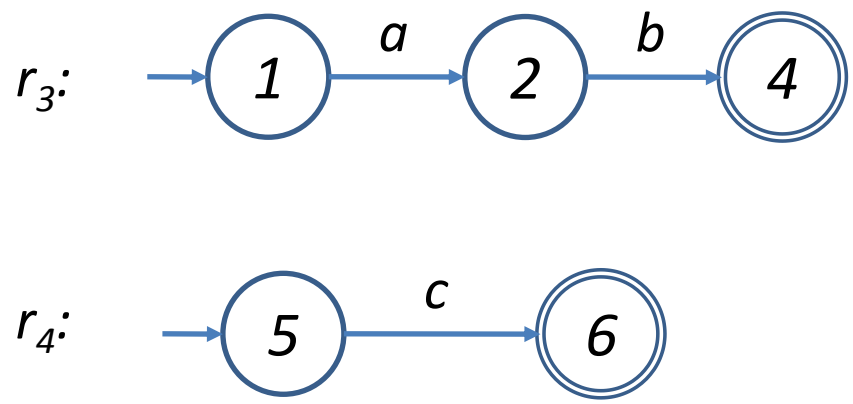
$$(\underbrace{(a \cdot b)}_{r_3} \mid c)^*$$

r_3:



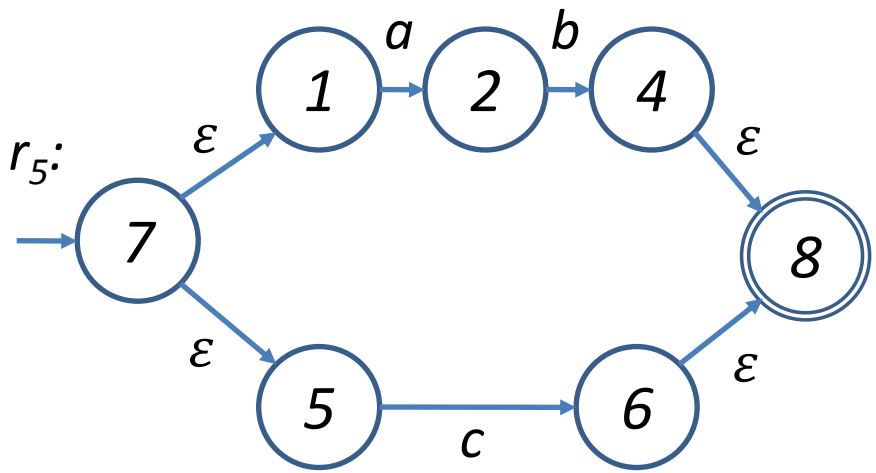
Step 4: construct NFA for r_4.

$$\Big(\underbrace{(a \cdot b)}_{r_3} \mid \underbrace{c}_{r_4}\Big)^*$$

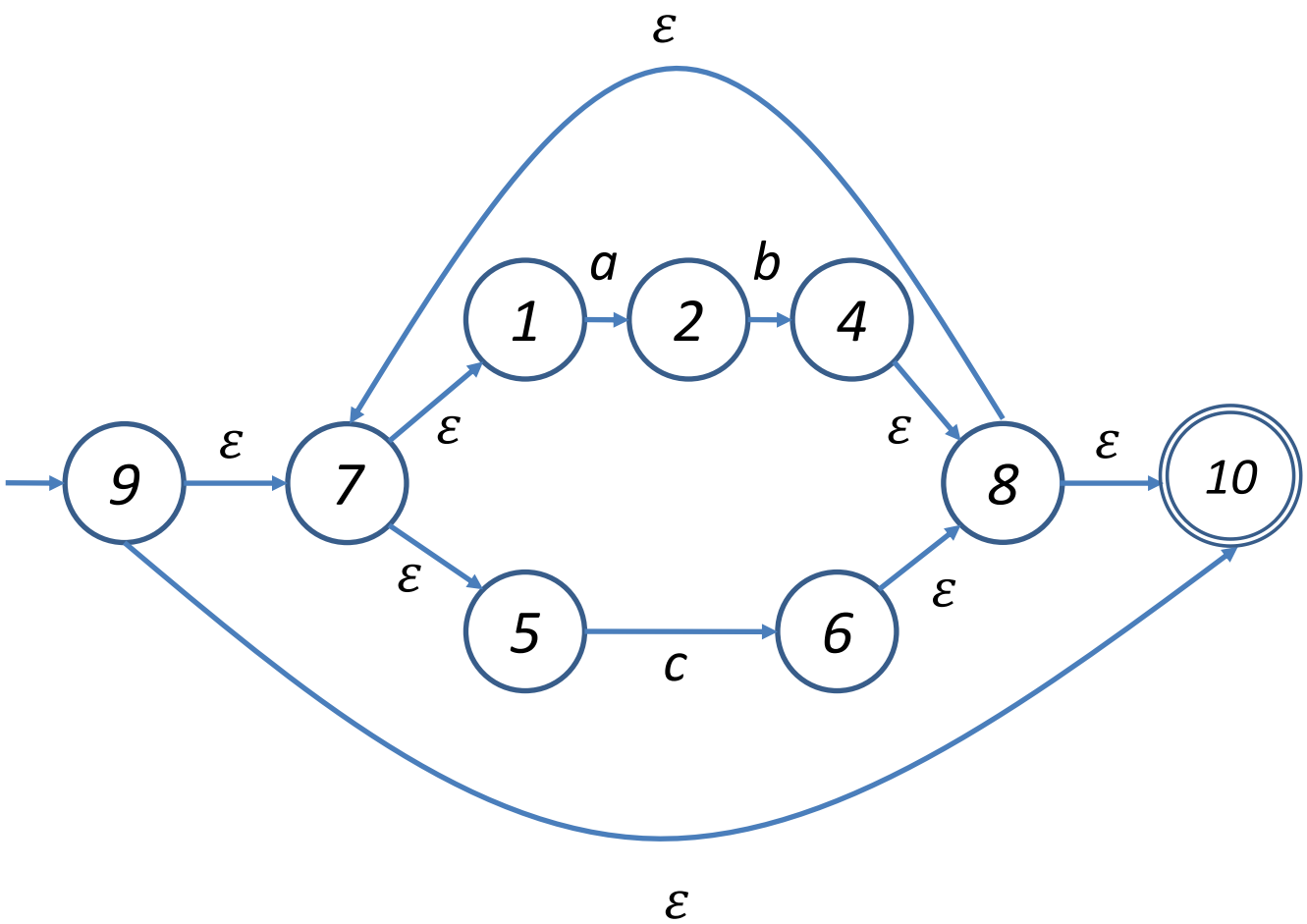


Step 5: construct NFA for r_5.

$$\Big(\underbrace{(a \cdot b) \mid c}_{r_5}\Big)^*$$



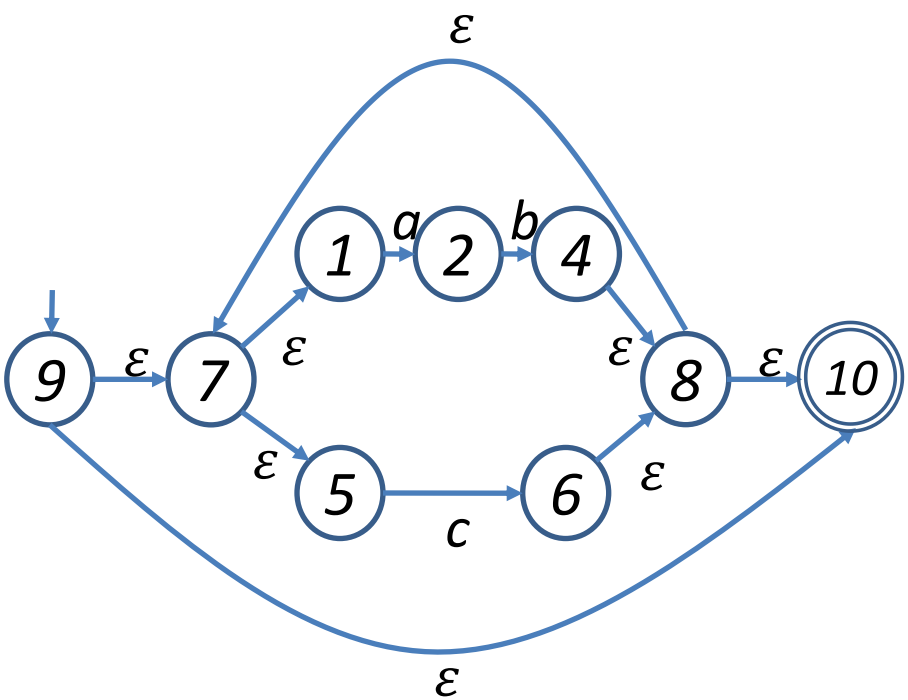
Step 6: construct NFA for r_5*.



SUBSET CONSTRUCTION

Convert the NFA to a DFA.

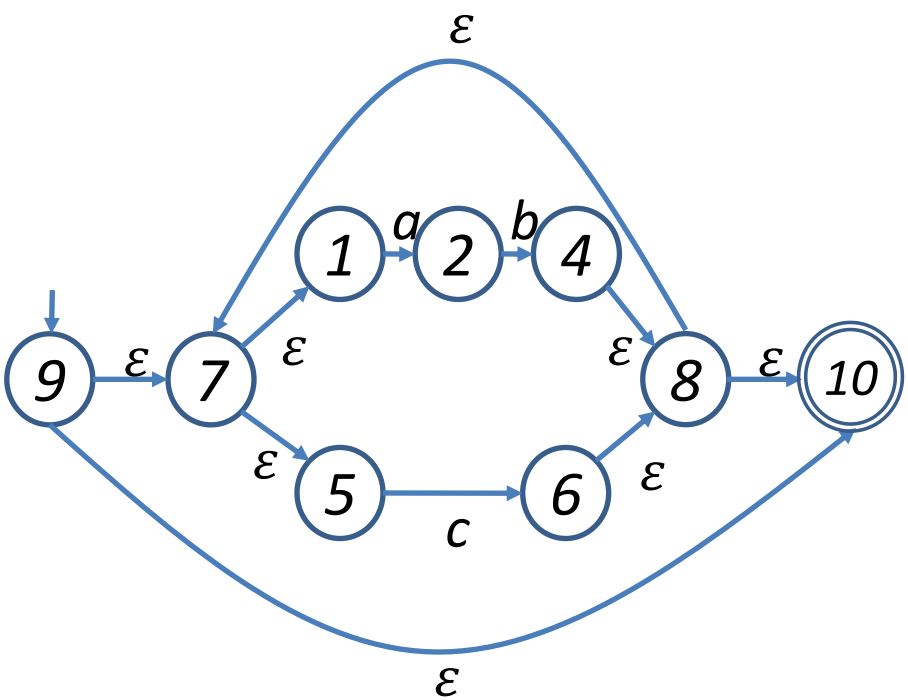
Draw transition table for DFA



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*

Add ε-*closure(9)* as DFA start state



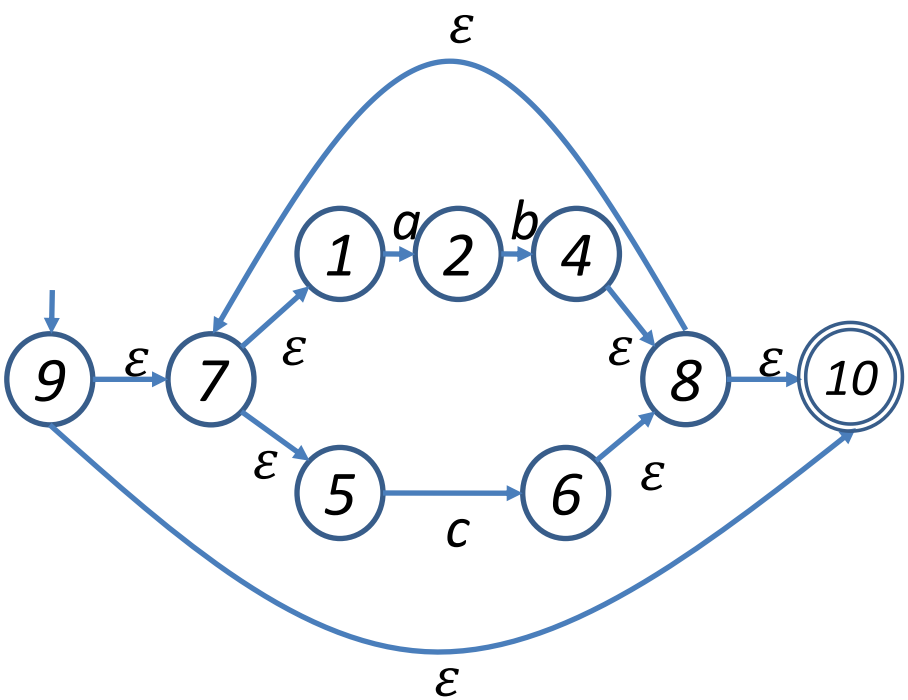
D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A			

Subset construction: **algorithm**

```
while (there is an unmarked state T in D_states) {
    mark T;
    for (each input symbol a) {
        U = ε-closure(move(T, a));
        D_tran[T, a] = U
        if (U is not in D_states)
            add U as unmarked state to D_states;
    }
}
```

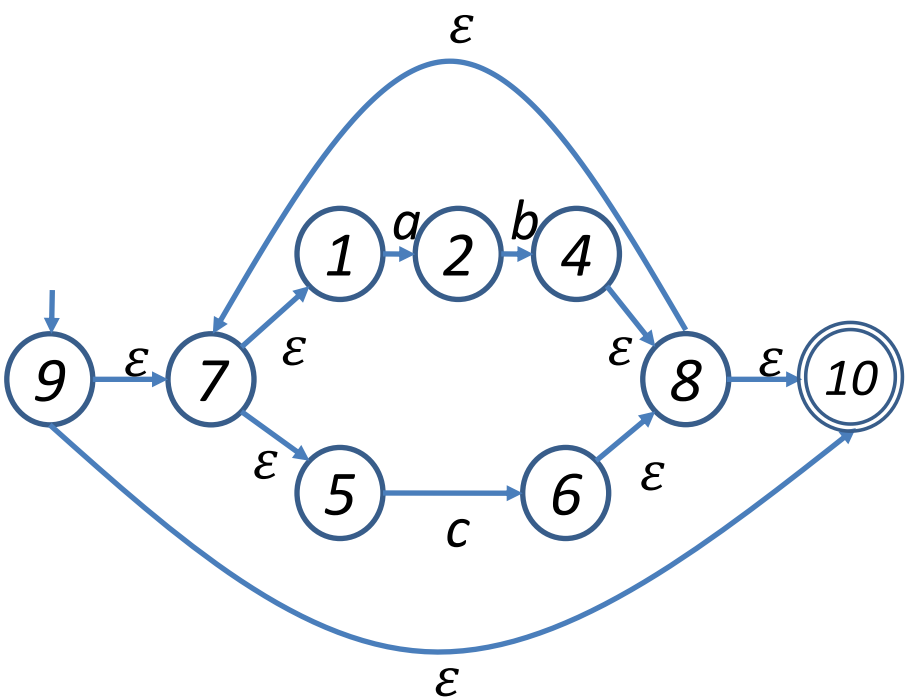
Mark state A



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓			

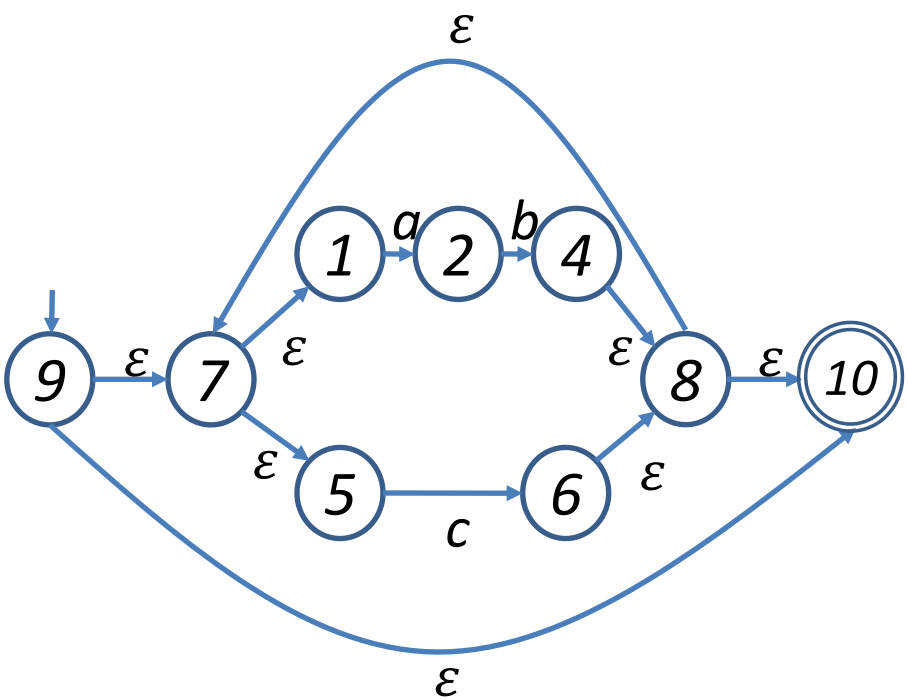
Compute ε-*closure(move(A, a))*



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B		
{2}	B			

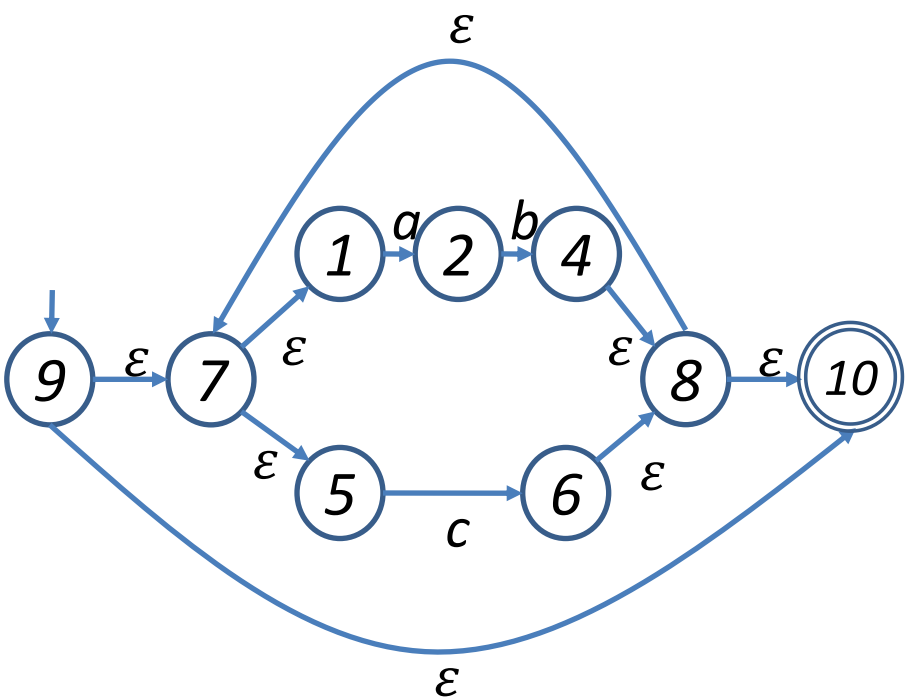
Compute ε-*closure(move(A, b))*



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	
{2}	B			

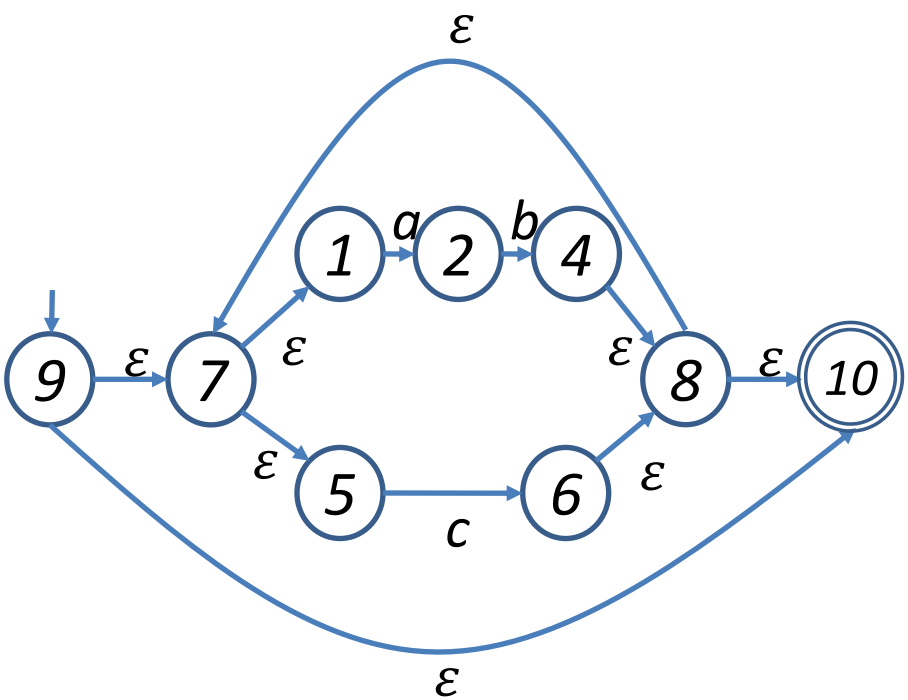
Compute ε-*closure(move(A, c))*



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B			
{6,8,10,7,1,5}	C			

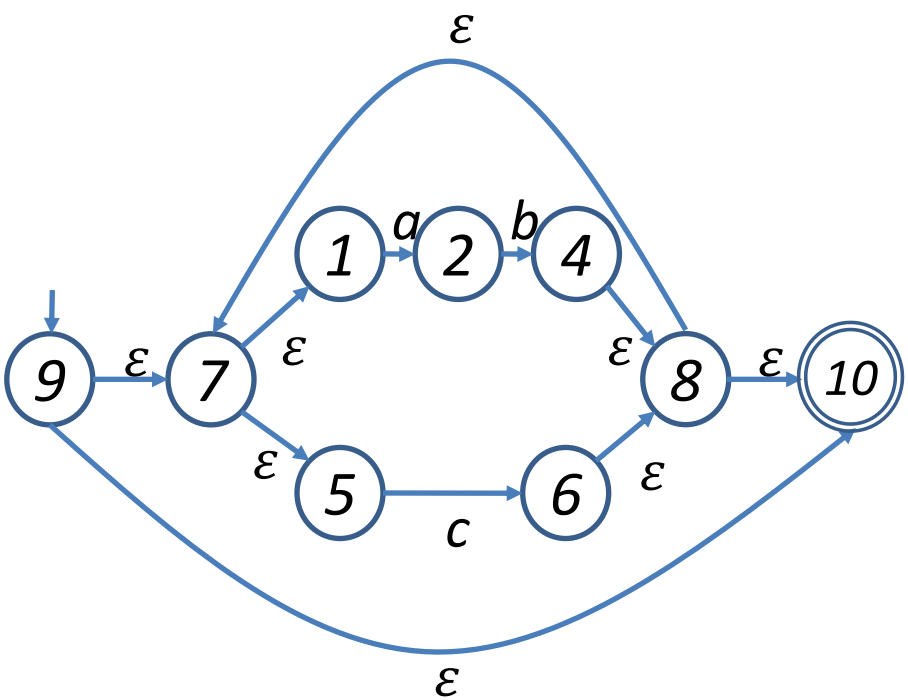
Mark B



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓			
{6,8,10,7,1,5}	C			

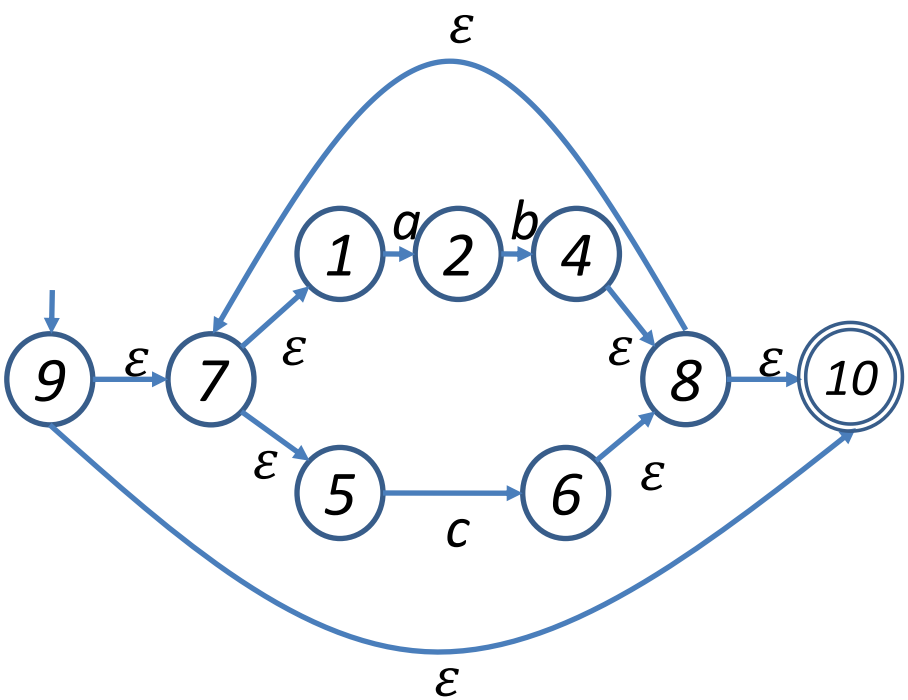
Compute ε-*closure(move(B, a))*



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-		
{6,8,10,7,1,5}	C			

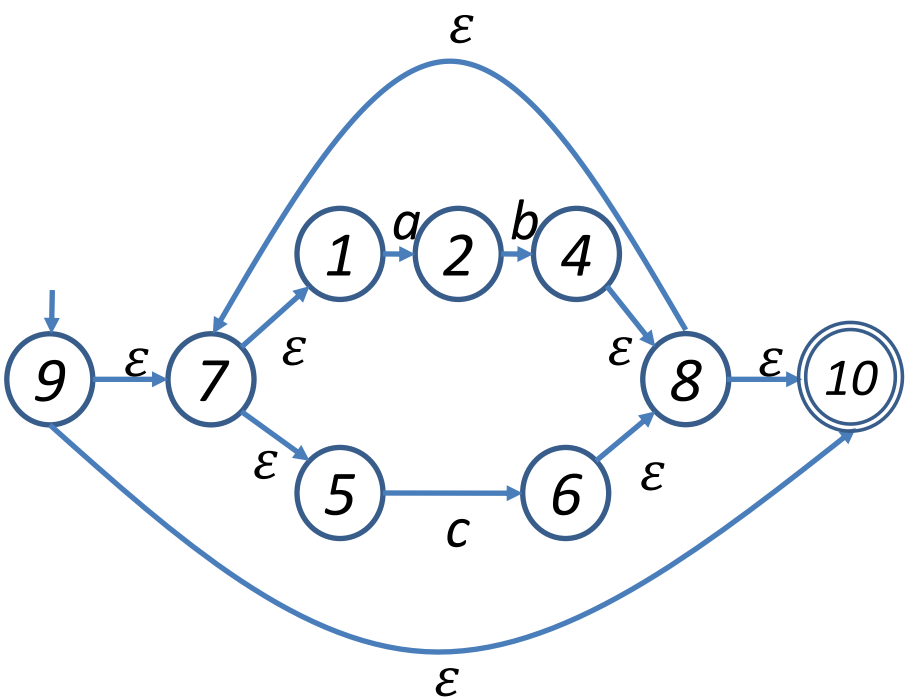
Compute ε-*closure(move(B, b))*



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-	D	
{6,8,10,7,1,5}	C			
{4,8,7,1,5,10}	D			

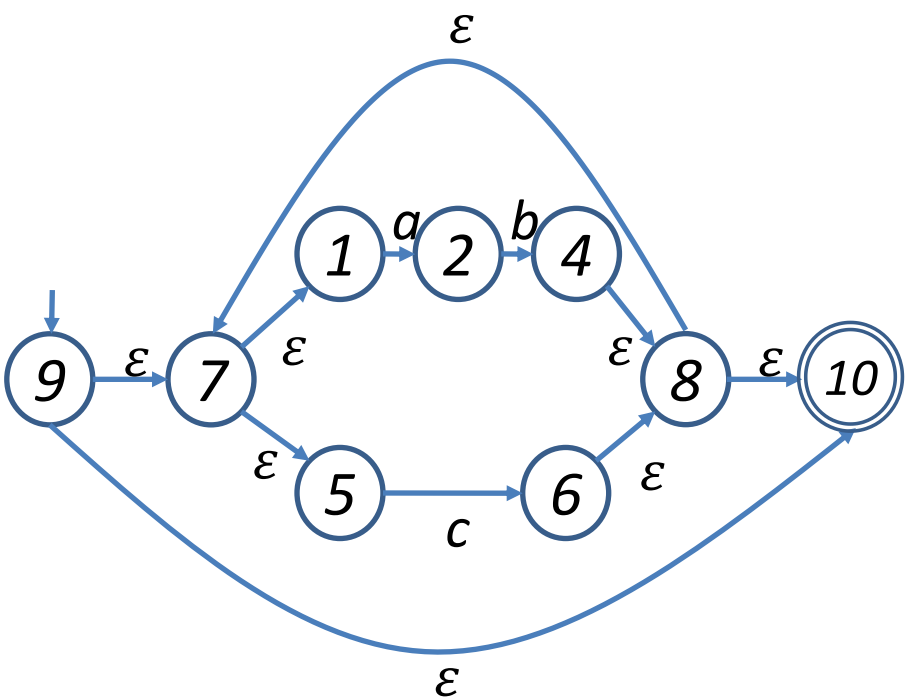
Compute ε-*closure(move(B, c))*



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-	D	-
{6,8,10,7,1,5}	C			
{4,8,7,1,5,10}	D			

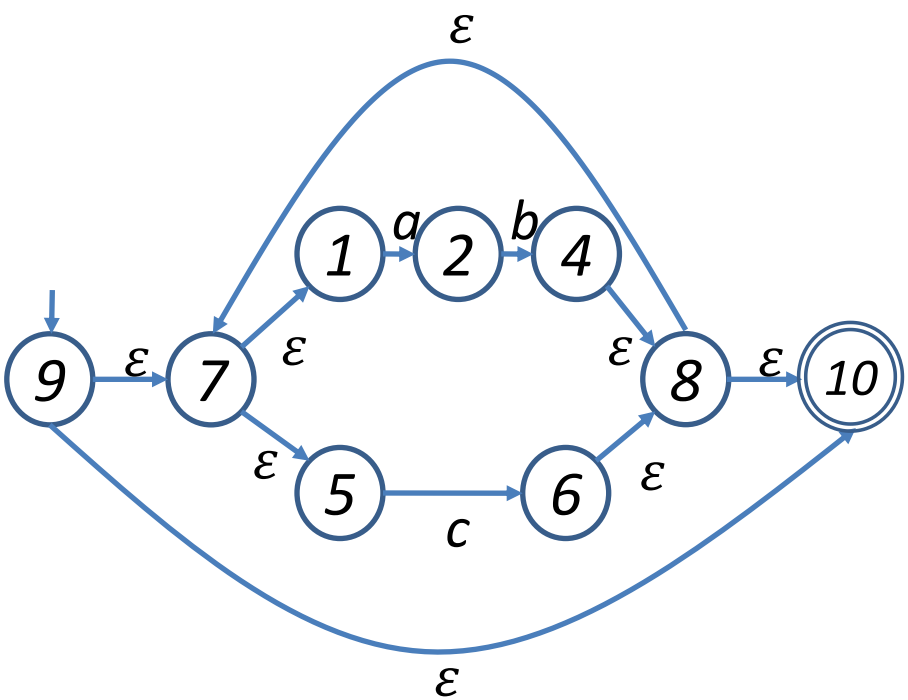
Mark C



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-	D	-
{6,8,10,7,1,5}	C ✓			
{4,8,7,1,5,10}	D			

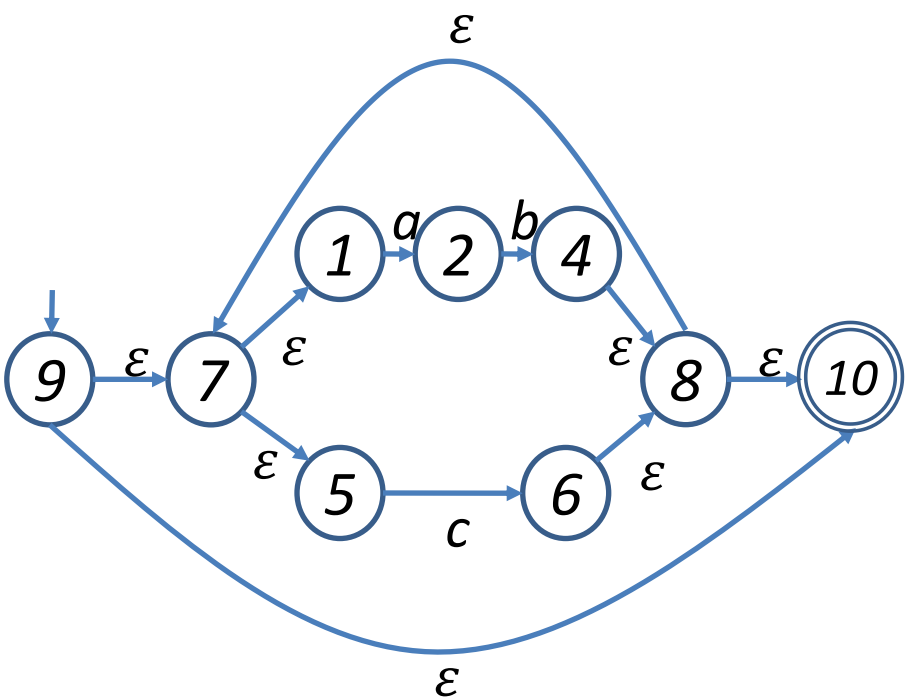
Compute ε-*closure(move(C, a))*



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-	D	-
{6,8,10,7,1,5}	C ✓	B		
{4,8,7,1,5,10}	D			

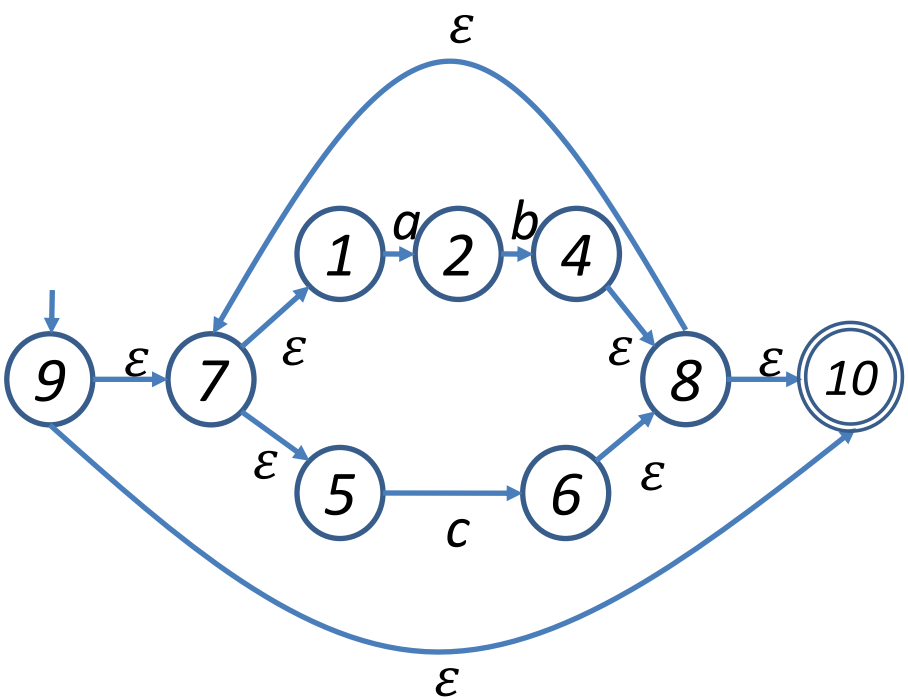
Compute ε-*closure(move(C, b))*



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-	D	-
{6,8,10,7,1,5}	C ✓	B	-	
{4,8,7,1,5,10}	D			

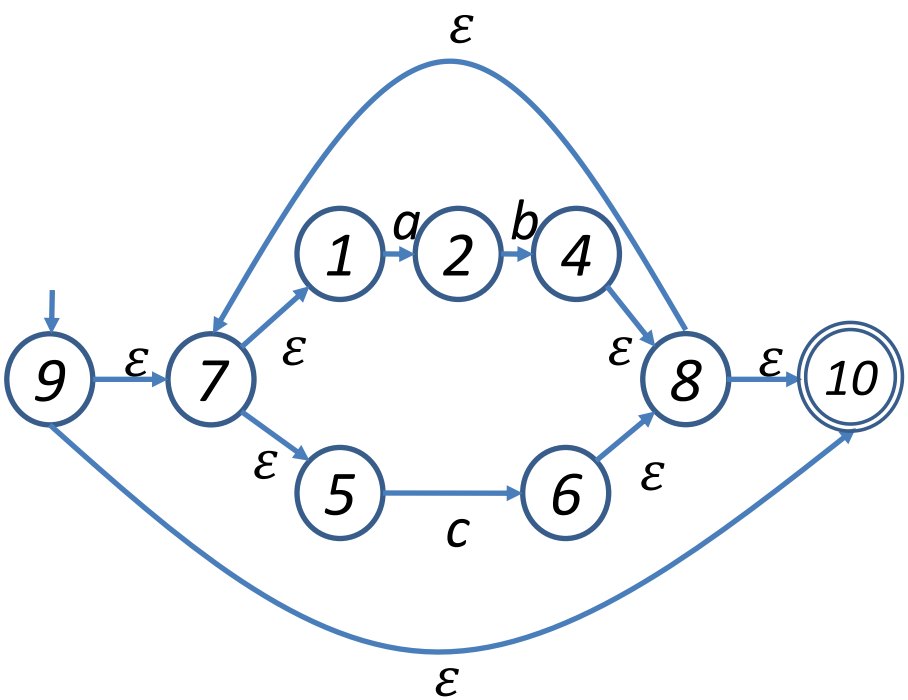
Compute ε-*closure(move(C, c))*



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-	D	-
{6,8,10,7,1,5}	C ✓	B	-	C
{4,8,7,1,5,10}	D			

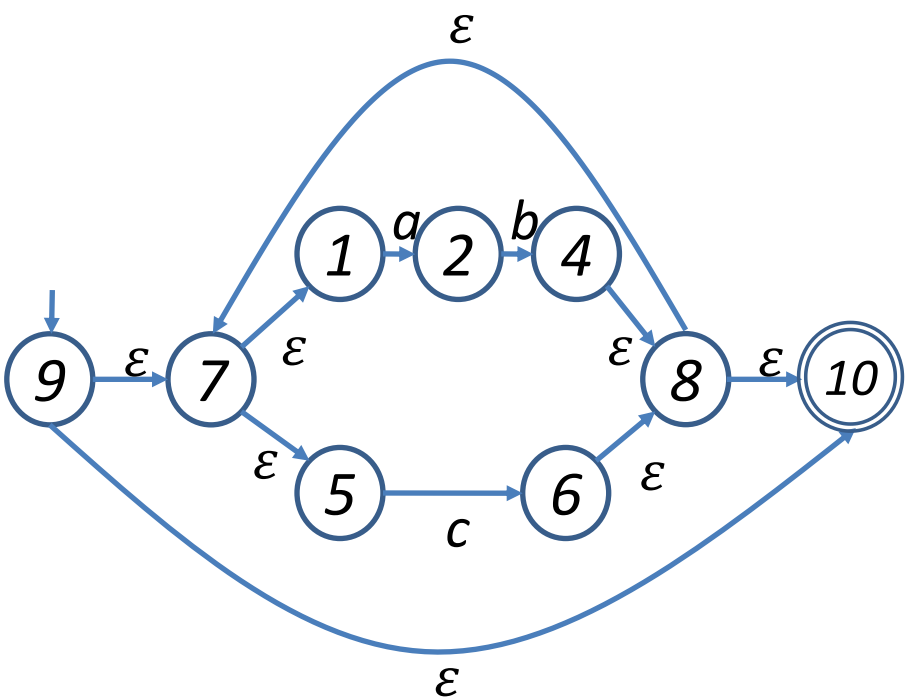
Mark D



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-	D	-
{6,8,10,7,1,5}	C ✓	B	-	C
{4,8,7,1,5,10}	D ✓			

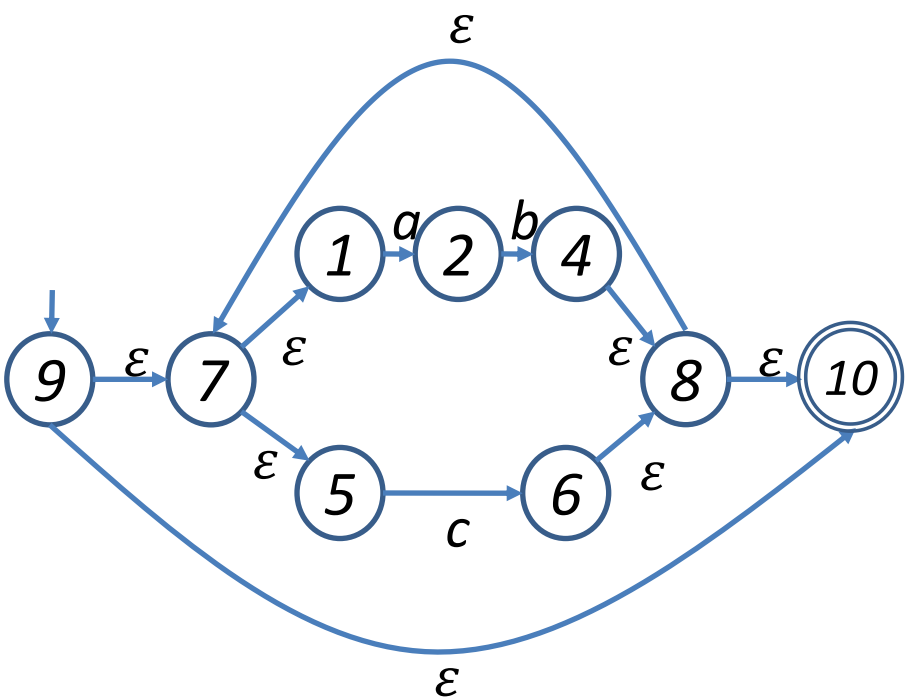
Compute ε-*closure(move(D, a))*



D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-	D	-
{6,8,10,7,1,5}	C ✓	B	-	C
{4,8,7,1,5,10}	D ✓	B		

Compute ε-*closure(move(D, b))*

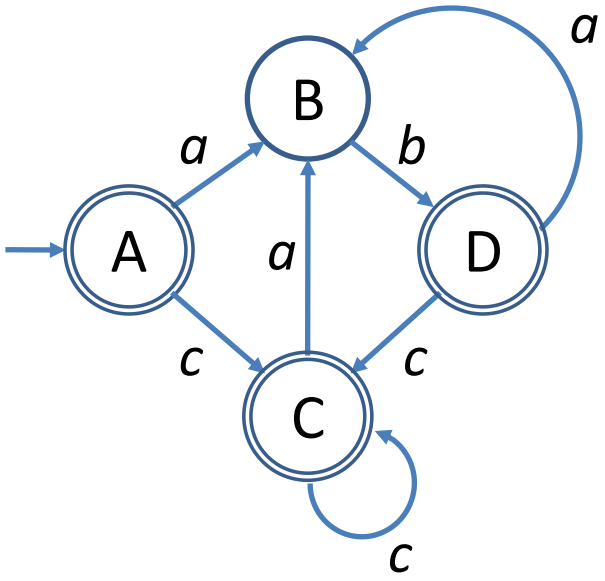


D_{states}

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-	D	-
{6,8,10,7,1,5}	C ✓	B	-	C
{4,8,7,1,5,10}	D ✓	B	-	C

Draw DFA

NFA States	DFA State	*Next State*		
		a	*b*	*c*
{9,7,1,5,10}	A ✓	B	-	C
{2}	B ✓	-	D	-
{6,8,10,7,1,5}	C ✓	B	-	C
{4,8,7,1,5,10}	D ✓	B	-	C



TRANSLATION TO C

Convert the DFA into C code.

```
int match(char* next)   {
   goto A;

   A:          if (*next == '\0') return 1;
               if (*next == 'a')  { next++; goto B; }
               if (*next == 'c')  { next++; goto C; }
               return 0;

   B:          if (*next == '\0') return 0;
               if (*next == 'b')  { next++; goto D; }
               return 0;

   C:          if (*next == '\0') return 1;
               if (*next == 'a')  { next++; goto B; }
               if (*next == 'c')  { next++; goto C; }
               return 0;

   D:          if (*next == '\0') return 1;
               if (*next == 'a')  { next++; goto B; }
               if (*next == 'c')  { next++; goto C; }
               return 0;
}
```