Compiler construction

Assignment 2

Donovan Schaafsma 13656007

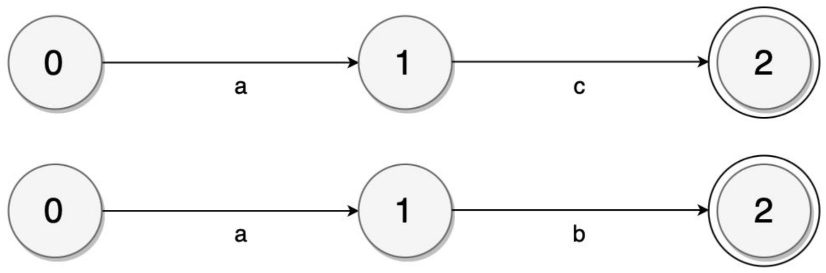
Tim van Ekert 13635565

Thompson’s Construction

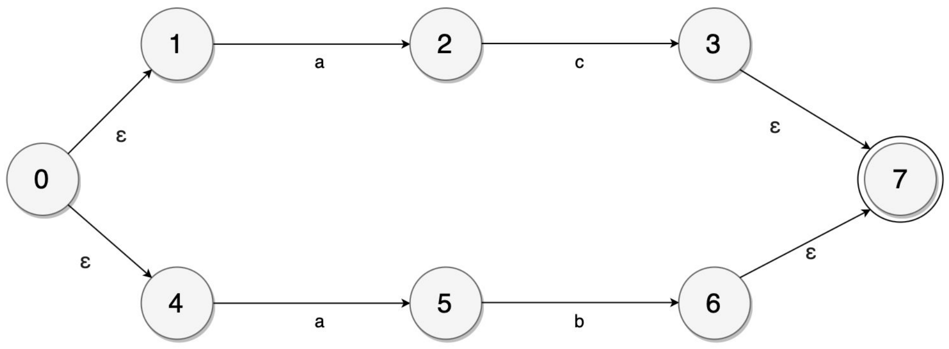
regex: (ac|ab)*

We are going to split up the regex in two small automatons. These two together will create the full NFA.

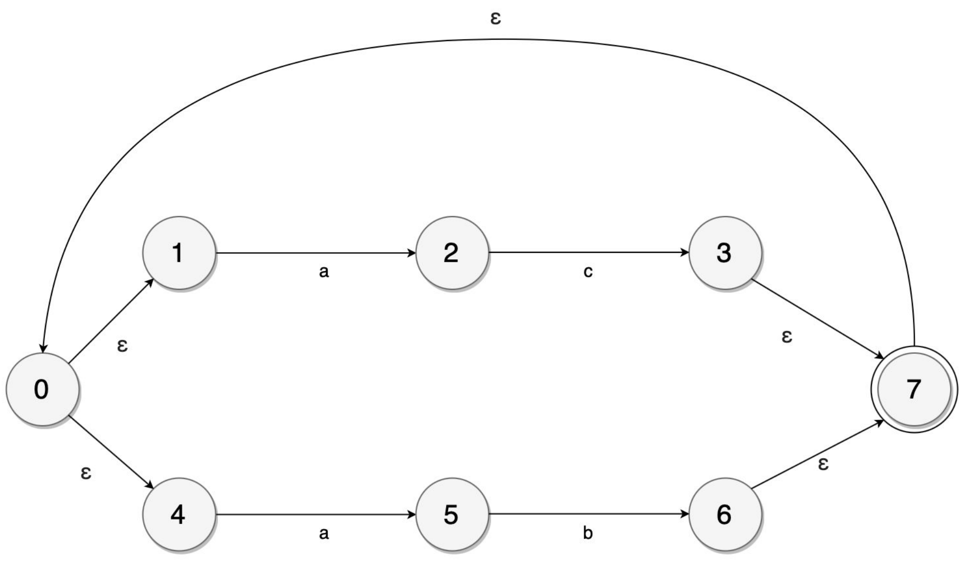
step 1:



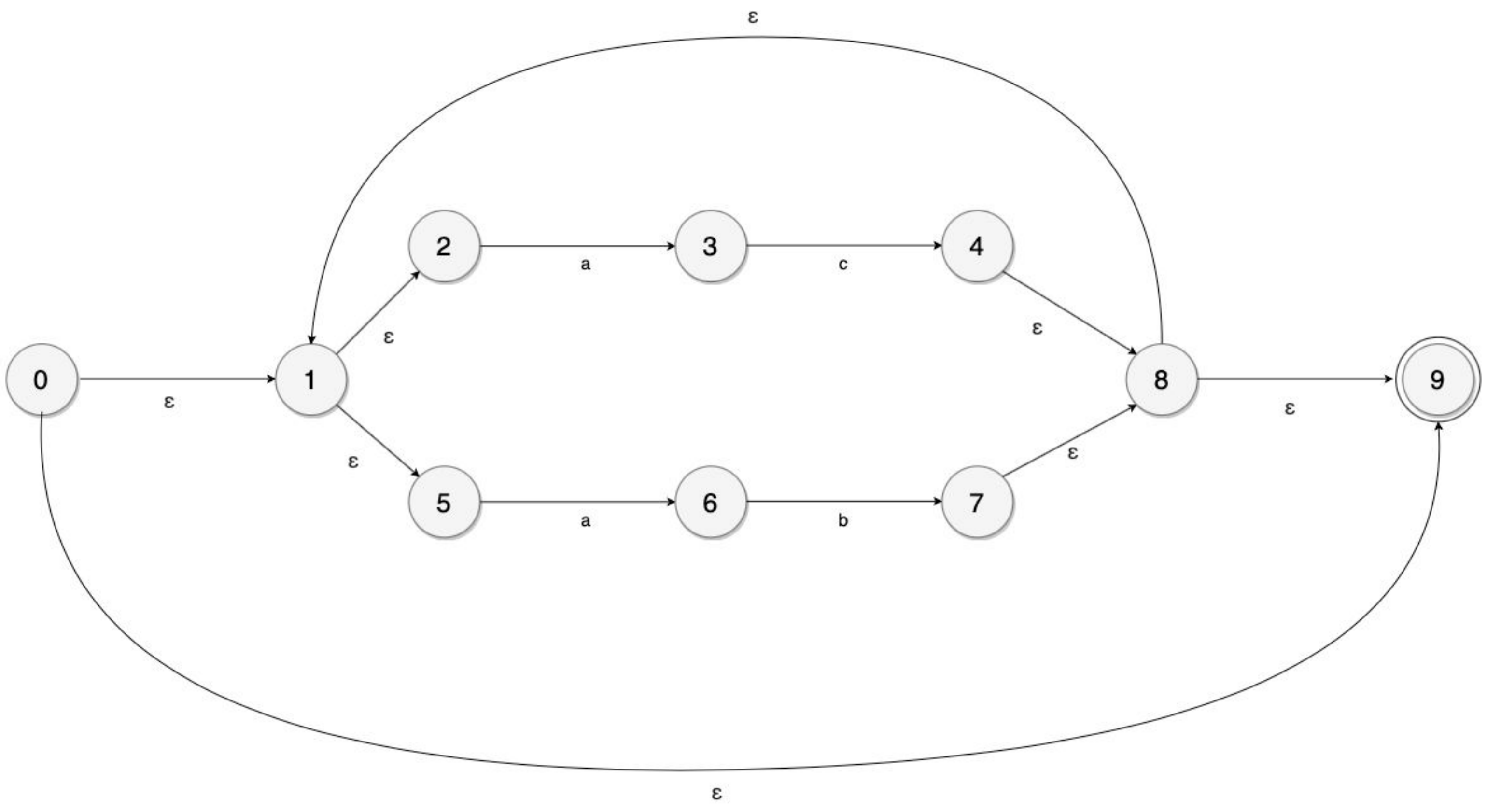
step 2:



step 3:



Final NFA:



Subset Construction

1. Get epsilon closures for each state by using the algorithm.

This is a table where we are checking the epsilon closures for every state.

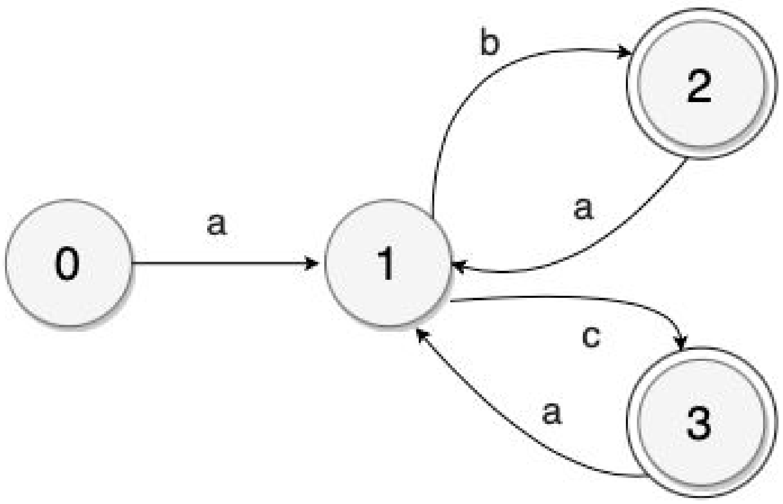
State	Epsilon Closures
0	{0, 1, 2, 5, 9}
1	{1, 2, 5}
2	{2}
3	{3}
4	{4, 8, 9, 1, 2, 5}
5	{5}
6	{6}
7	{7, 8, 9, 1, 2, 5}
8	{8, 9, 1}
9	{9}

2. Transition table

This is the transition table based on following the DFA algorithm.

NFA State	DFA State		a	b	c
{0, 1, 2, 5, 9}	0		{3, 6}	-	-
{3, 6}	1		-	{7,8,9,1,2,5}	{4,8,9,1,2,5}
{7,8,9,1,2,5}	2		{3, 6}	-	-
{4,8,9,1,2,5}	3		{3, 6}	-	-

3. DFA diagram



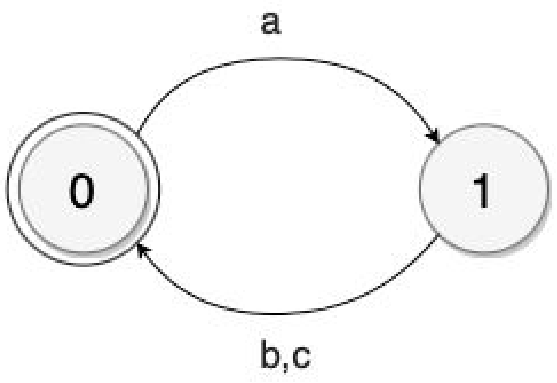
Hopcroft's Algorithm

1. Transition table

After splitting up the DFA table, we have found the next minimised-DFA states documented in the table.

DFA State	min-DFA State		a	b	c
{0, 2, 3}	0		{1}		
{1}	1			{0, 2, 3}	{0, 2, 3}

2. Minimised DFA diagram



Direct-coded Scanner

This scanner is built based on the minimised DFA diagram in C++.

```
/**
*  Scanner for regex (ac|ab)*
*/
char *scanner(char *stream) {
   int pos = 0; char c;

// scan and check for character a
state_init:
   c = stream[pos++];
   if (c == 'a')
       goto state_1;
   if (pos == 1)
       goto state_err;
   else
       goto state_succ;

// scan and check for character a
state_0:
   c = stream[pos++];
   if (c == 'a')
       goto state_1;
   else
       goto state_succ;

// scan and check for character b or c
state_1:
   c = stream[pos++];
   if (c == 'c' || c == 'b')
       goto state_0;
   else
       goto state_err;

// state success return stream
state_succ:
   return stream;



// state error so return null
state_err:
   return NULL;
}
```