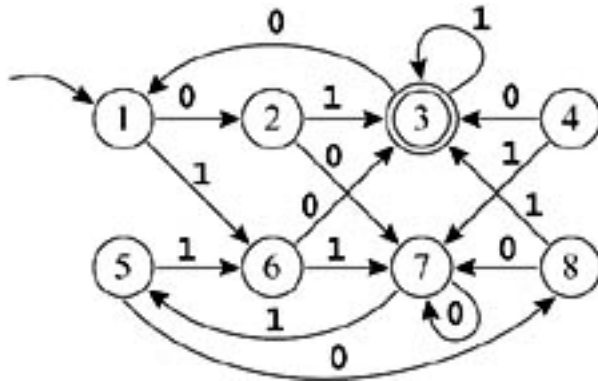


## 1. DFA Minimisation

Minimise the DFA given below, following the algorithm presented in the lecture.



Vi må først identificere vores exit states, og derefter gruppere herefter.

g1: { 3 }

g2: { 1 , 2 , 4 , 5 , 6 , 7 , 8 }

Nu kan vi benytte algoritmen til yderligere at opdele grupperingen:

	0	1
1	g2	g2
2	g2	g1
4	g1	g2
5	g2	g2
6	g1	g2
7	g2	g2
8	g2	g1

g1: { 3 }

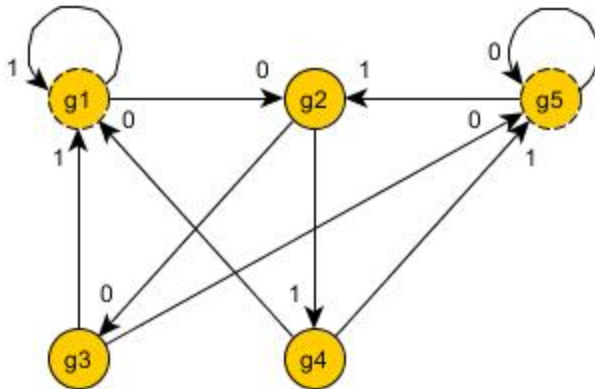
g2: { 1 , 5 }

g3: { 2 , 8 }

g4: { 4 , 6 }

g5: { 7 }

Vi kan nu tegne vores minimerede DFA og få:



Her er de stiplede linjer exit-states, og g1 er vores entry-state.

## 2. Backtracking Automaton

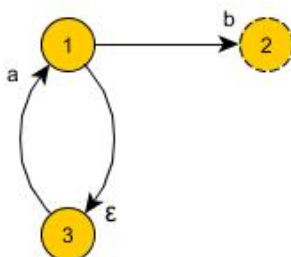
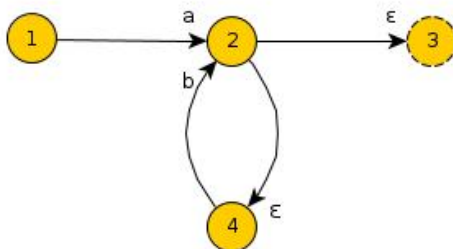
Given the following regular expressions.

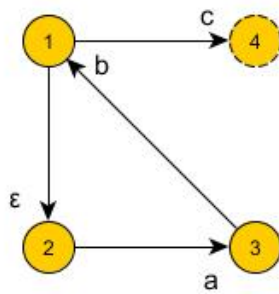
$$\alpha_1 = ab^*$$

$$\alpha_2 = a^*b$$

$$\alpha_3 = (ab)^*c$$

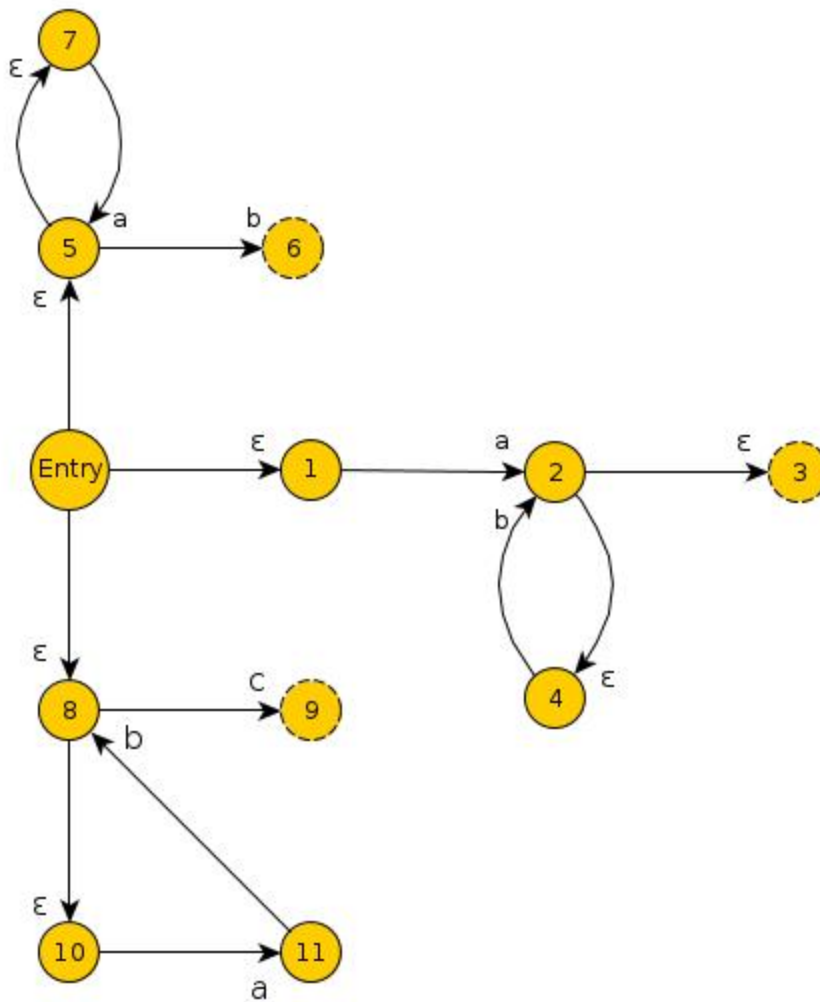
(a) Give an NFA for each expression (no particular method required), and construct a combined NFA to recognise  $L(\alpha_1)$ ,  $L(\alpha_2)$ , and  $L(\alpha_3)$ .





Igen er de stiplede linjer exit states, og i alle tilfælde er 1 vores entry state.

Vi kan konstruere vores kombinerede NFA, ved trivielt at sammensætte vores NFA'er beskrevet ovenfor:



(b) Convert this combined NFA to a DFA by subset construction.

Vi starter med  $s_0$  som er vores epsilon enclosure  $s_0 = \{1, 2, 6, 7, 9, 10\}$ .

Nu kan vi kigge på move for henholdsvis a, b og c.

$\text{move}(s_0, a) = \{2, 3, 4, 5, 7, 11\}$

$\text{move}(s_0, b) = \{6\}$

$\text{move}(s_0, c) = \{9\}$

Nu kan vi lave  $s_1$  ved blot at udelukke  $\{6\}$  og  $\{9\}$  fordi de er færdige.

$s_1 = \{2, 3, 4, 5, 7, 11\}$

$\text{move}(s_1, a) = \{5, 7\}$

$\text{move}(s_1, b) = \{2, 4, 6, 8\}$

$\text{move}(s_1, c) = \{\}$

Vi gentager for  $s_2 = \{5, 7\}$  og  $s_3 = \{2, 4, 6, 8\}$

$\text{move}(s_2, a) = \{5, 7\}$

$\text{move}(s_2, b) = \{6\}$

$\text{move}(s_2, c) = \{\}$

$\text{move}(s_3, a) = \{10, 11\}$

$\text{move}(s_3, b) = \{2, 4\}$

$\text{move}(s_3, c) = \{9\}$

Vi gentager for  $s_4 = \{10, 11\}$  og  $s_5 = \{2, 4\}$

$\text{move}(s_4, a) = \{11\}$

$\text{move}(s_4, b) = \{8\}$

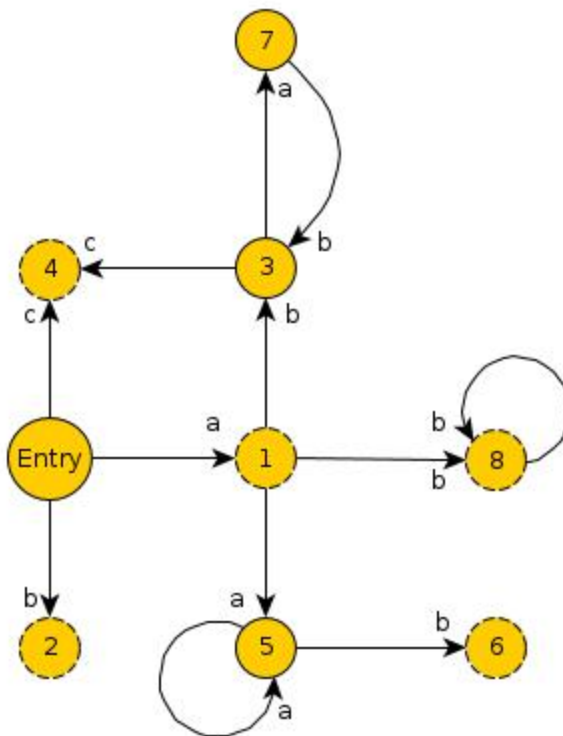
$\text{move}(s_4, c) = \{\}$

$\text{move}(s_5, a) = \{\}$

$\text{move}(s_5, b) = \{2, 4\}$

$\text{move}(s_5, c) = \{\}$

Nu kan vi så konstruere vores DFA.



(c) Describe the transitions and backtracking steps used when the combined DFA analyses the input "ababacca"

Vi tager følgende steps: 1 → 8 backtrack til 1 → 3 → 7 backtrack til entry → 4 backtrack til entry → 4 backtrack til entry → 1.

### 3. Mosmllex Scanner Construction

Use Mosmllex to implement a scanner for the regular expressions  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$  from task 1.2.

Use the generated scanner to verify your solution of part 1.2 c).

Den genererede scanner giver det forventede output. Giver vi den aaabbbbaaa får vi:

```
val it = [Symbol "aaab", Symbol "b", Symbol "b", Symbol "a", Symbol "a", Symbol "a",
  EOF] : TokenType list
```

### 4. More on Regular Languages and Tokenisation

(a) Using the alphabet of decimal digits, give regular expression describing the following languages:

i. Numbers divisible by 5.

$(0|5)|([0-9]^+0|5)$

ii. Numbers in which digit '5' occurs exactly three times.

$[^5]^*5[^5]^*5[^5]^*5[^5]^*$

Hvilket også kan skrives en smule flottere som:  $([^5]^*5)\{1,3\}[^5]^*$

*(b) Are the following languages over the alphabet of decimal digits regular? Glve short convincing reasons for your answers.*

*i. Numbers which contain digit '1' exactly as many times as digit '2'.*

Regulære sprog er sprog som kan beskrives fuldstændigt. Eftersom dette sprog indeholder en uendelig mængde tal, er det ikke muligt for os at lave en komplet beskrivelse.

*ii. Numbers  $N < 1.000.000$  which contain digit '1' exactly as often as digit '2'.*

I modsætning til ovenstående eksempel, har vi nu en øvre grænse for antallet af tal/ord i sproget, og vi kan derfor lave en komplet beskrivelse. Sproget er altså regulært.