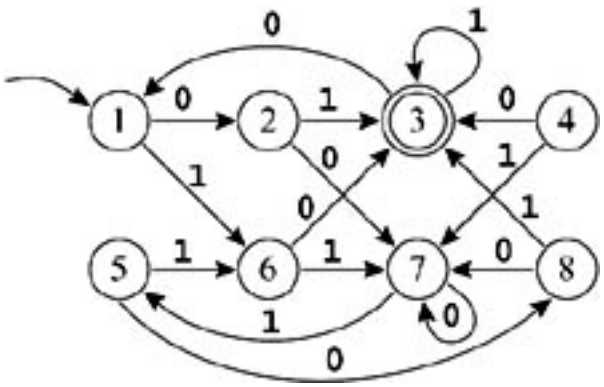
1. DFA Minimisation

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 }

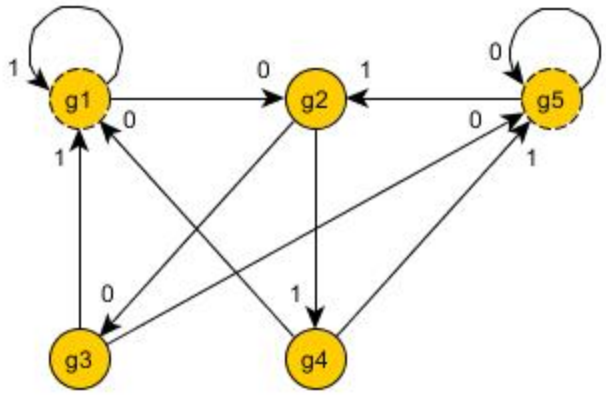
Nu kan vi benytte algoritmen til yderligere at opdele grupperingen:

	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

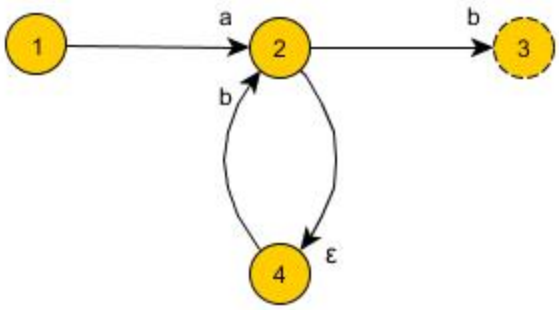
Given the following regular expressions.

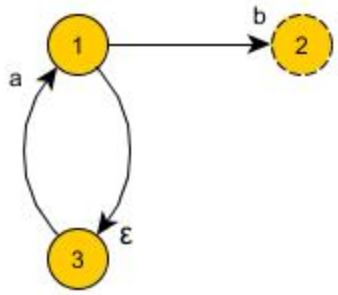
$\alpha 1 = ab*$

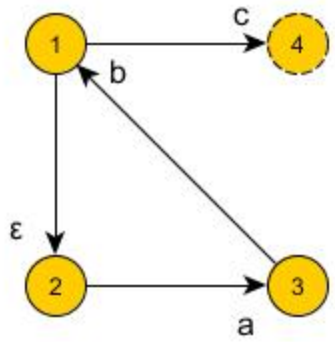
$\alpha 2 = a*b$

$\alpha 3 = (ab)*c$

(a) Give an NFA for each expression (no particular method required), and construct a combined NFA to recognise L(α1), L(α2), and L(α3).







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

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

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

3. Mosmllex Scanner Construction

Use Mosmllex to implement a scanner for the regular expressions α1, α2, and α3 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 *aaabbbaaa* 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.

`5|([0-9]+[05])`

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? GIve 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.