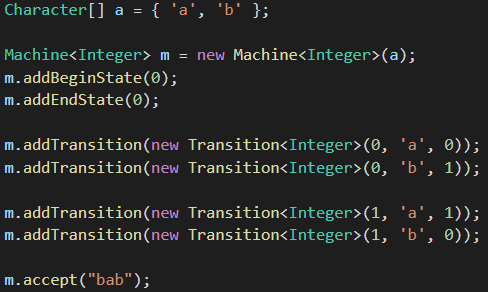
Formele methode Ties & Chiem

Ties Tienhoven (2143936) & Chiem Stevens (2135167)

# Applicatie beschrijving

Voor het vak formele methode hebben wij een applicatie gerealiseerd die verschillende onderdelen van automaten theorie implementeert, dit hebben wij in Java ontwikkeld. De volgende aspecten zijn gerealiseerd:

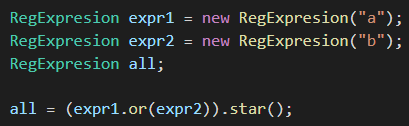
## DFA / NDFA:

Voor het representeren van een DFA of NDFA hebben wij een Machine klasse ontwikkeld. Deze maakt gebruik van sets om begin en eind staten bij te houden en een lijst aan transities om de overgangen tussen staten op te slaan. Met deze klasse kunnen bijvoorbeeld woorden geaccepteerd worden naar de mee gegeven regels en kan alle taal worden gegenereerd die bij deze regels hoort. Ook kan deze geconverteerd worden van NDFA naar DFA en kan de DFA geminimaliseerd worden.

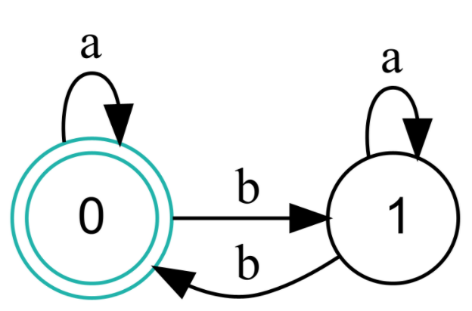
Voor deze Machine klasse is ook een Transition klasse ontwikkeld die een overgang tussen twee staten representeert. Deze transitie kan een karakter of epsilon bevatten die gebruikt worden om een overgang te accepteren, of een reguliere expressie bevatten om toegepast te kunnen worden voor de thompson conversie.

## RegExpression & RegExParser

Voor het representeren van een reguliere expressie hebben we gebruik gemaakt van de bestaande voorbeeld code, deze hebben we uitgebreid om te kunnen kijken of twee reguliere expressies aan elkaar gelijk zijn en om de thompson conversie toe te passen om een reguliere expressie om te zetten naar een NDFA.

Hierbij hebben we een parser ontwikkeld die een reguliere expressie als string kan omzetten naar een representatie van deze reguliere expressie gebruikmakend van de RegExpression klasse.

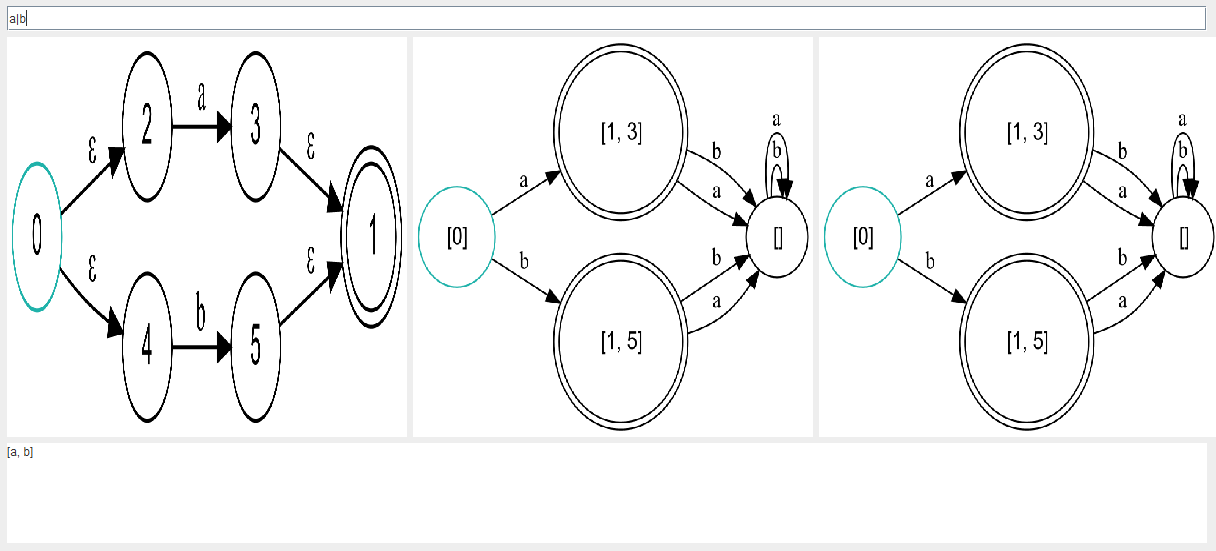
## GraphizGenerator

Voor het tekenen van DFA’s/NDFA’s hebben we een GraphizGenerator klasse ontwikkeld die Graphiz plaatjes kan genereren naar een gegeven machine.



## User interface

Tenslotte hebben we een simpel user interface ontworpen wat een gegeven reguliere expressie omzet naar een NDFA, deze NDFA wordt omgezet naar een DFA waarna deze wordt geminimaliseerd. Hiernaast wordt aan de onderkant afgebeeld welke woorden deze reguliere expressie accepteert.



# Gerealiseerde onderdelen

|  |  |  |
| --- | --- | --- |
| Onderdeel | Gerealiseerd | Plaats in code |
| Hardcoded voorbeelden van (N)DFA en regulier expressies | Ja | Test klassen in test folder:   * MachineTest.java * RegexTest.java |
| Parser voor reguliere expressies | Ja | RegExParser klasse met voorbeeld code in RegexParserTest.java |
| Woorden genereren aan de hand van de ingevoerde reguliere expressie  Woorden genereren die niet in de taal zitten | Ja  Nee | getLanguage functie in RegExpression klasse (RegExpression.java) |
| Voor DFA: testen of een woord wel/niet geaccepteerd wordt | Ja | accept functie in machine klasse (Machine.java) |
| Constructor operaties voor DFA om snel test DFA's te maken:   * begint met xyz * eindigt op xyz * bevat xyz | Ja | MachineCreator klasse (MachineCreator.java) met de methodes:   * startsWith * endsWith * contains |
| Operaties op DFA (liefst met tupel-constructie):   * en * of * niet | Nee |  |
| Thompson constructie | Ja | thompsonConvert functie in RegExpression klasse (RegExpression.java) |
| NDFA->DFA  Zonder en met epsilon | Ja met epsilon | toDFA functie in Machine klasse (Machine.java) |
| Minimalisatie  (via reverse)  Bonus: ook algoritme uit het dictaat (via opsplitsen tabel) | Ja  Ja | reverse functie in Machine klasse (Machine.java)  minimize functie in Machine klasse (Machine.java) |
| Gelijkheid op reguliere expressies / (NDFA) | Ja op reguliere expressie | isEqualTo functie in RegExpression klasse (RegExpression.java) |
| TestApplicatie:  (Makkelijk testen, plaatje met GraphViz, wegschrijven test resultaten…) | Ja, Simpel UI en mogelijkheid om plaatjes te genereren van DFA of NDFA | Simpel UI via MainScreen klasse (MainScreen.java)  Genereren van plaatjes via draw of drawName functie in Machine klasse (Machine.java) |