Practicum 2:
Reguliere Expressies en Eindige Automaten

Contactinformatie:
Adres *ELIS (verdieping -3), Technicum (blok I), Sint-Pietersnieuwstraat 41*
E-mail *compiler@elis.ugent.be*

BELANGRIJK: Het indienen van de oplossingen dient steeds te gebeuren via het Dropbox-menu op Minerva, dit zowel voor een tijdelijke versie op het einde van elk practicum als de finale versie voor onderstaande dealine. Gebruik het `make_tarball.sh` script om een archief te genereren, en zorg dat het naar de begeleiders verzonden wordt.

DEADLINE: 3 maart, 23:59.

Het is de bedoeling dat dit practicum door iedereen **individueel** wordt uitgevoerd, m.a.w. iedere student van iedere groep moet zijn eigen oplossing indienen.

1 Doelstelling

In dit practicum is het de bedoeling om voor een gegeven reguliere expressie een automaat te construeren gebruik makend van *JFLAP*. JFLAP is een grafisch hulpmiddel om eenvoudig en snel de basisconcepten van formele talen en automaten te leren. Meer informatie is te vinden op `http://www.jflap.org`. De automaten die je construeert sla je op zoals aangegeven in Sectie 2. **Hou je aan deze naamconventies**! Om fouten te voorkomen is het aangeraden de bestanden die reeds aanwezig zijn bij de initiële programmabestanden verder aan te vullen.

2 Opgave

Gegeven de volgende lexicale specificatie:

```
token A: (ab*|a)
token B: (a(b*)a)?
token C: (a?|b+)
```

1. Construeer voor elk van deze tokens een NFA die het token kan herkennen. Gebruik hiervoor strikt het algoritme beschreven in Hoofdstuk 2 van de cursus. Elke automaat sla je respectievelijk op als `token_a.jff`, `token_b.jff` en `token_c.jff`. Met behulp van de `step with closure` functionaliteit die je vindt onder het menu `Input` in JFLAP kan je nagaan welke toestanden je automaat doorloopt voor een bepaalde input en of de automaat de input al dan niet aanvaardt.

2. Voeg deze automaten samen tot één grote automaat die alle drie de tokens herkent. Hou de aanvaardingstoestanden voor de drie tokens echter gescheiden, zodat uit de aanvaardingstoestand die bereikt wordt nog steeds kan afgeleid worden welk token juist herkend werd. Sla deze automaat op als `nfa.jff`. Om de hoeveelheid werk voor de volgende vraag te beperken is het aan te raden de automaat te optimaliseren door *indien mogelijk* verschillende opeenvolgende λ-bogen samen te nemen.

3. Construeer voor deze NFA de equivalente DFA volgens het algoritme in hoofdstuk 2 van de cursus. Schrijf de afleiding van de verschillende toestanden van de DFA neer in een tabel zoals Tabel 1.

   Sla deze tabel op in het bestand `nfa2dfa.csv`, dat je kan vinden bij de initiële practicumbestanden. Teken de resulterende automaat en sla deze op als `dfa.jff`.

4. Geef voor elke aanvaardingstoestand van de DFA aan welk token hij herkent. Hanteer hiervoor volgende regels:

   - Er wordt steeds een zo lang mogelijke karakterstring gematcht.
   - Voor twee karakterstrings van gelijke lengte, krijgt het eerder gespecificeerde token voorrang. Met andere woorden: token A heeft voorrang op token B, dat op zijn beurt weer voorrang heeft op token C.

   Geef aan welke toestanden de DFA doorloopt voor de inputstring `abaabbaba`. Welke tokens zal de lexicale analyser herkennen? Sla je antwoord op in een bestand *dfa-answer.txt*.

toestand D_x	input vb. a	$edge(D_x, a)$	$DFAedge(D_x, a)$	nieuwe toestand D_y
0	ε	...	...	D_0
D_0	a	...	...	D_1

Table 1: Toestand- en overgangstabel van een eindige automaat.