Deeltentamen Grammatica's en ontleden 16 december 2004

Let op: opgave 1 t/m 4 tellen voor (slechts) 5 punten mee, opgave 5 en 6 ieder voor (maar liefst) 40.
In opgave 1 t/m 4 staan steeds drie beweringen. Eén daarvan is nonsens; de uitspraak is zo onzinnig dat je aan het vaststellen van de waarheid niet eens toekomt: er zit bij wijze van spreken een typeringsfout in. Van de andere twee uitspraken is de ene waar, en de andere onwaar. Geef bij elke opgave aan

- welke uitspraak nonsens is, en vertel in 1 zin waarom
- welke uitspraak waar is: toon dat kort aan, of geef een tegenvoorbeeld voor de onware uitspraak

1. a. Het complement van een reguliere grammatica is ook regulier

   b. Het complement van een reguliere taal is ook regulier

   c. Het complement van een contextvrije taal is ook contextvrij

2. a. In een niet-ambigue grammatica heeft elke zin precies één *derivation*

   b. In een ambigue grammatica zijn er zinnen met meer dan één *leftmost derivation*

   c. In een niet-ambigue grammatica zijn er geen zinnen met meer dan één *leftrecursive derivation*

3. Stel G is een contextvrije grammatica (T, N, R, S). Dan geldt:

   a. $L(G) \;=\; \{w \mid T^* \stackrel{*}{\Rightarrow} w \;,\; w \in S\}$

   b. $L(G) \;=\; \{w \mid S \stackrel{*}{\Rightarrow} w \;,\; w \in T^*\}$

   c. $L(G) \;=\; \{w \mid S \rightarrow w \in R \;,\; w \in T^*\}$

4. (in deze opgave betekent 'nonsens': fout getypeerd). In de parser-combinator library geldt:

   a. `succeed n =  const n  <$> epsilon`

   b. `succeed n =        n  <$> epsilon`

   c. `succeed n = (\x -> n) <$> failp`

5. a. Geef het type van de Parser-combinators `<|>`, `<*>` en `<$>`

   b. Gegeven zijn twee parsers `p` en `q` met onderstaand type,
   en een parser `r` die daar gebruik van maakt:

```
p :: Parser Char a
q :: Parser Char b
r =  f <$> p <*> q
```

   Kan `f` hierin een functie zijn met één parameter? Met twee parameters? Met drie parameters? (Je moet dus drie aparte antwoorden geven). Geef voor elk van de drie gevallen aan:
   (als het wel kan) wat zijn de voorwaarden aan het type van `f`, en wat is het type van `r` ?
   (als het niet kan) waarom niet?

   c. Gegeven zijn de basis-parsers en de parser-combinators `<|>`, `<*>`, `<$>` uit de library (maar niet de extra utilities zoals many, many1, listOf, option, chainr). Schrijf met behulp daarvan nu zelf de parser-combinator

```
many1 :: Parser a b -> Parser a [b]
```

   die overeenkomt met de '+' constructie uit EBNF.

(zie achterkant voor vervolg)

d. We bekijken de taal van boolean expressies in Java (iets vereenvoudigd).
Deze expressies zijn opgebouwd uit:
- de constanten `True` en `False`
- variabelen, zoals `p` en `stop`
- functies met nul of meer parameters, waarbij als parameters uitsluitend integer constanten worden gebruikt, zoals `f(1,2,3)` en `even(37)` en `foo()`
- de logische operatoren `&&`, `||` en `!`, waarbij `&&` een hogere prioriteit heeft dan `||`, en `!` de hoogste prioriteit heeft.
- haakjes voor groepering

Definieer een datatype `Expr`, die ontleedbomen van expressies in deze taal beschrijft.

e. Definieer voor deze taal een parser

```
parseExpr :: Parser Char Expr
```

Je mag daarbij alle parser-combinators uit de library gebruiken.
Ook mag je gebruik maken van

```
identifier :: Parser Char String
integer    :: Parser Char Int
```

Je mag ook extra hulpfuncties en -parsers schrijven.
Je hoeft geen rekening te houden met spaties of andere whitespace.

6. In deze opgave werken we met alfabet $\{a, b\}$. Gegeven zijn de reguliere grammatica C met nonterminals $\{P, Q, S\}$, startsymbool S, en regels:

$$
\begin{aligned}
S &\to a\,P \\
S &\to b\,Q \\
P &\to \varepsilon \\
Q &\to a
\end{aligned}
$$

en de reguliere grammatica D met nonterminals $\{T, U, V, W, X, Y\}$, startsymbool T, en regels:

$$
\begin{aligned}
T &\to b\,T \\
T &\to a\,W \\
T &\to W \\
W &\to a\,X \\
W &\to b\,Y \\
X &\to a\,U \\
Y &\to a\,b \\
U &\to \varepsilon \\
U &\to a\,U
\end{aligned}
$$

a. Geef een reguliere grammatica G zodat $L(G) = L(C)\,L(D)$, dus strings in de taal van G zijn de concatenatie van een string uit de taal van C en een string uit de taal van D.

b. Construeer een Niet-deterministische Finite-state Automaton (NFA) met hetzelfde alfabet en dezelfde taal als G. Je mag kiezen of je de automaat tekent, of alle mogelijkheden van de overgangsfunctie opsomt. Geef in ieder geval duidelijk de start- en eind-toestand(en) aan. (Hint: maak de grammatica eerst 'zeer regulier').

c. Construeer een Deterministische Finite-state Automaton (DFA) met hetzelfde alfabet en dezelfde taal als G. (Ook hier mag je weer tekenen of opsommen).