UITWERKING tentamen Grammatica's en Ontleden, 27 januari 2005

De cursieve tekst vormt geen deel van de antwoorden, maar slechts een toelichting daarop.

1. (a)
```
type PredAlgebra c
      = ( String -> String     -> c
        , String -> String     -> c
        , c      -> c          -> c
        , c      -> c          -> c
        , c      -> c          -> c
        , String -> Range -> c -> c
        , String -> Range -> c -> c
        )
```
(b)
```
foldPred :: PredAlgebra c -> Pred -> c
foldPred fs@(f1,_,_,_,_,_,_) (Eq  x y)   = f1 x y
foldPred fs@(_,f2,_,_,_,_,_) (NEq x y)   = f2 x y
foldPred fs@(_,_,f3,_,_,_,_) (And p q)   = f3 (foldPred fs p) (foldPred fs q)
foldPred fs@(_,_,_,f4,_,_,_) (Or  p q)   = f4 (foldPred fs p) (foldPred fs q)
foldPred fs@(_,_,_,_,f5,_,_) (Imp p q)   = f5 (foldPred fs p) (foldPred fs q)
foldPred fs@(_,_,_,_,_,f6,_) (All x u p) = f6 x u (foldPred fs p)
foldPred fs@(_,_,_,_,_,_,f7) (Exi x u p) = f7 x u (foldPred fs p)
```
of korter:
```
foldPred fs@(f1,f2,f3,f4,f5,f6,f7) = fold
 where fold (Eq  x y)   = f1 x y
       fold (NEq x y)   = f2 x y
       fold (And p q)   = f3 (fold p) (fold q)
       fold (Or  p q)   = f4 (fold p) (fold q)
       fold (Imp p q)   = f5 (fold p) (fold q)
       fold (All x u p) = f6 x u (fold p)
       fold (Exi x u p) = f7 x u (fold p)
```
(c)
```
freeAlgebra :: PredAlgebra [String]
freeAlgebra = ( \x y   -> [x,y]
              , \x y   -> [x,y]
              , \p q   -> p ++ q
              , \p q   -> p ++ q
              , \p q   -> p ++ q
              , \x u p -> filter (/=x) p
              , \x u p -> filter (/=x) p
              )
```
of korter:
```
freeAlgebra = (f, f, g, g, h, h, h)
     where f x y   = [x,y]
           g p q   = p++q
           h x u p = filter (/=x) p
```
(d)
```
negAlgebra :: PredAlgebra Pred
negAlgebra = ( \x y   -> NEq x y
             , \x y   -> Eq  x y
             , \p q   -> Or  p q
             , \p q   -> And p q
             , \p q   -> Imp q p       -- fout
             , \x u p -> Exi x u p
             , \x u p -> All x u p
             )
```
Of korter:
```
negAlgebra = (Neq, Eq, Or, And, flip Imp, Exi, All)
```

(e)
```
evalAlgebra :: PredAlgebra (Env -> Bool)
evalAlgebra = ( \x y   -> \e ->  e ? x == e ? y
              , \x y   -> \e ->  e ? x /= e ? y
              , \p q   -> \e ->  p e && q e
              , \p q   -> \e ->  p e || q e
              , \p q   -> \e ->  not(p e) || q e
              , \x u p -> \e ->  and [p ((x,v):e) | v <- u]
              , \x u p -> \e ->  or  [p ((x,v):e) | v <- u]
              )
eval :: Pred -> Bool
eval p = foldPred evalAlgebra  p []
```

2. (a) Bewijs van stelling 3: De reguliere taal L wordt beschreven door een toestandsautomaat A. De n waarvan het bestaan in de stelling geclaimd wordt is het aantal toestanden van A. Een (deel)zin z van de taal correspondeert met een pad in A. Als de lengte van zo'n deelzin $\geq n$ is, is er minstens één toestand waar je tweemaal langskomt. Laat w het stuk van de zin zijn dat correspondeert met zo'n lus in het pad. In plaats van 1 keer kun je dit stuk ook 0 of ≥ 2 keer inlassen, waarbij start- en eindtoestand niet veranderen. Dus uvw^ixy blijft een zin van L voor alle i.

   (b) Bekijk de taal $L = \{\mathsf{a}^n\mathsf{b}^n \mid n \in \mathbb{N}\}$.
   - L is een contextvrije taal, want hij wordt beschreven door een grammatica met regel $S \to \varepsilon \mid \mathsf{a}S\mathsf{b}$.
   - L is geen reguliere taal. Dit blijkt uit toepassen van stelling 4: laat n gegeven zijn. Neem $u = \varepsilon$, $z = \mathsf{a}^n$ en $x = \mathsf{b}^n$. Iedere opsplitsing daarvan bestaat uitsluitend uit a's. Neem $i = 2$. Omdat $|w| > 0$, is $|vw^ix| > |vwx| = n$. Dus uvw^ix bevat meer dan n symbolen a, en x precies n symbolen b. Dus $uvw^ixy \notin L$.

   (c) Ja, de klasse van reguliere talen is gesloten onder complementatie. Zij A een automaat die een reguliere taal L herkent. Maak de toestandsovergang-functie van A zonodig totaal door een extra 'sink state' toe te voegen. De automaat waarin alle eind-toestanden van A juist geen eind-toestand meer zijn en omgekeerd is nu een automaat die $\overline{L}$ herkent, dus $\overline{L}$ is ook weer regulier.

   (d) Ja, de klasse van contextvrije talen is gesloten onder vereniging. Laten twee grammatica's voor twee contextvrije talen gegeven zijn. Maak de nonterminal symbolen hiervan uniek door zonodig systematische naamwijziging in één van de grammatica'. Voeg de regels nu samen, en voeg twee extra regels toe: $S \to S'$ en $S \to S''$, waarbij S een nieuw startsymbool is, en S' en S'' de oorspronkelijke. Deze grammatica beschrijft de vereniging van de oorspronkelijke talen.

3. (a) Je kunt de lookaheadset bepalen van een *regel* van een grammatica. Zo'n lookaheadset is een *verzameling van terminal symbolen*.

   (b) Om een (deel)zin horend bij een nonterminal symbool te herkennen probeerden de parser-combinators *alle* alternatieven toe te passen. Sommige daarvan leveren misschien geen oplossing op; dat blijkt dan later vanzelf. Bij LL(1)-ontleden wordt echter *vooraf* het te gebruiken alternatief gekozen, namelijk dat waarvan de lookaheadset het eerstvolgende input-symbool bevat.

   (c) De lookaheadsets voor alle regels van een nonterminal moeten onderling disjunct zijn.

   (d)
```
run []     []                = true
run []     (x:xs)            = false
run (a:as) []                = false
run (a:as) (x:xs) | isLower a = a==x && run as xs
                  | isUpper a = run (rs++as) (x:xs)
    where rs      = snd (fst (filter ok gram))
          ok rule = a==fst rule && elem x (lookaheadset rule)
```

   (e) Het alfabet van de automaat bestaat uit de symbolen (nonterminal èn terminal) van de grammatica. De toestanden zijn de 'items' van de grammatica, dat wil zeggen regels met een cursorpositie in de rechterkant van de regel. De stackmachine gebruikt de automaat om te kiezen uit 'shift' (een inputsymbool op de stack zetten) en 'reduce' (een rechterkant van een regel herkennen op de stack en vervangen door de bijbehorende nonterminal).