TENTAMEN Grammatica's en Ontleden, 29 januari 2004, 14–17 uur

1. Het volgende Haskell datatype definieert ontleedbomen voor een bepaald soort rekenkundige expressies:

```
data Expr = Con Int
          | Var Char
          | Add Expr Expr
          | Mul Expr Expr
          | Sigma Char Expr Expr Expr
```

In de expressies komen constanten en variabelen voor, en optellingen en vermenigvuldigingen. En als vijfde mogelijkheid kunnen *sommaties* worden weergegeven. Bijvoorbeeld,

```
Sigma 'j' (Con 0) (Con 5) (Mul (Var 'x') (Var 'j'))
```

is de ontleedboom van de expressie die in de wiskunde meestal wordt opgeschreven als

$$\sum_{j=0}^{5} x \times j$$

De drie expressies duiden de ondergrens, de bovengrens en de body van de sommatie aan. Merk op dat in zo'n sommatie een variabele wordt gebonden (in het voorbeeld j), die in de body kan worden gebruikt. De binding geldt alleen voor de body, niet voor de onder- en bovengrens.

(a) Definieer een Haskell `type` voor een bij dit type expressies horende algebra.

(b) Geef het *type* van de functie `foldExpr`, waarmee aan de hand van een algebra een ontleedboom recursief kan worden doorlopen. Dus vul in:

```
foldExpr :: ....
```

Je hoeft de functie zelf dus niet te definiëren. In de rest van deze opgave mag je aannemen dat de functie bestaat.

(c) Definieer een algebra `free` waarmee alle vrije variabelen van een expressie kunnen worden bepaald (in een lijst, waarbij het niet erg is als die dubbelen bevat), en geef ook het type daarvan. Dus vul in:

```
free :: ....
free  = ....
```

In het voorbeeld hierboven is x een vrije variabele, omdat die niet door een Σ wordt gebonden; j is niet een vrije variabele, omdat hij alleen maar wordt gebruikt in de body van een Σ met j als tellende variabele.

(d) Definieer een algebra `eval` waarmee de waarde van een expressie kan worden bepaald. Je mag daarbij aannemen dat de expressie geen vrije variabelen bevat. Geef ook van deze algebra bovendien het type.

(e) Gegeven is een met parser-combinators gemaakte `Expr`-parser

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

Schrijf met behulp daarvan en van de algebra's (*die je ook mag gebruiken als je het vorige onderdeel hebt overgeslagen!)* een functie

```
value :: String -> Maybe Int
```

die een string als expressie ontleedt, en vervolgens van de expressie de waarde bepaalt. Als er bij het ontleden syntaxfouten worden ontdekt, als de zin ambigu is, of als er vrije variabelen in de expressie voorkomen moet `value` de waarde `No` opleveren, anders `Yes` toegepast op de berekende waarde.

Hierbij is `Maybe` het standaard Haskell datatype

```
data Maybe a = Yes a
             | No
```

2. In een contextvrije taal L geldt:

er bestaan	c, d	$: c, d \in \mathbb{N}$	:
voor alle	z	$: z \in L$ en $\lvert z\rvert > c$	:
er bestaan	u, v, w, x, y	$: z = uvwxy$ en $\lvert vx\rvert > 0$ en $\lvert vwx\rvert \leq d$	:
voor alle	$i \in \mathbb{N}$	$: uv^i wx^i y \in L$	

   (a) Meestal wordt deze stelling in zijn contrapositieve vorm gebruikt.
   Formuleer hoe de stelling dan luidt.

   (b) Bewijs met behulp van deze stelling dat $\{\mathsf{a}^n\mathsf{b}^n\mathsf{c}^n \mid n \in \mathbb{N}\}$ geen contextvrije taal is.

   (c) Is de klasse van contextvrije talen gesloten onder vereniging (d.w.z.: is de vereniging van twee contextvrije talen weer een contextvrije taal)? Geef een bewijs van het antwoord!

   (d) Is de klasse van contextvrije talen gesloten onder doorsnede?
   Geef een bewijs van het antwoord!

3. Gegeven is de volgende grammatica (hoofdletters zijn nonterminals, kleine letters zijn terminals, ε is de lege string, S is het startsymbool; voor het gemak zijn de productieregels genummerd):

1.	S	$\rightarrow$	$A\ B$
2.	S	$\rightarrow$	$\mathsf{r}\ A\ \mathsf{w}$
3.	A	$\rightarrow$	$D\ E$
4.	B	$\rightarrow$	p
5.	B	$\rightarrow$	$\mathsf{q}\ C$
6.	C	$\rightarrow$	$E\ \mathsf{z}$
7.	C	$\rightarrow$	y
8.	D	$\rightarrow$	ε
9.	D	$\rightarrow$	y
10.	E	$\rightarrow$	ε
11.	E	$\rightarrow$	x

   Van elke nonterminal kunnen de eigenschap *empty* en verzamelingen terminals *firsts* en *follow* worden bepaald.

   (a) Voor elke nonterminal N is de eigenschap *empty* gedefinieerd door

   $$\mathit{empty}(N) \;=\; N \stackrel{*}{\Rightarrow} \varepsilon$$

   Definieer nu, ook met gebruikmaking van $\stackrel{*}{\Rightarrow}$, de verzameling $\mathit{firsts}(N)$, zoals nodig in de $LL(1)$-analyse.

   (b) Vul in onderstaande tabel de kolommen *empty* en *firsts* in
   (# duidt het einde van de input aan):

nonterminal	*empty*	*firsts*	*follow*
S			$\{\#\}$
A			$\{\mathsf{p}, \mathsf{q}, \mathsf{w}\}$
B			$\{\#\}$
C			$\{\#\}$
D			$\{\mathsf{w}, \mathsf{x}, \mathsf{z}\}$
E			$\{\mathsf{w}, \mathsf{z}\}$

   (c) Bepaal van alle elf productieregels de *lookahead set*.

   (d) Bij $LL(1)$-ontleden voert een stackmachine steeds een aktie uit op grond van het symbool op de top van de stack. Op welke manier zijn de *lookahead sets* daarbij nodig?

   (e) Bij $LR(1)$-ontleden heeft de stackmachine bij elke stap de keus tussen een *shift* en een *reduce*. Wat houdt een *shift* in? En wat houdt een *reduce* in?
   (Je hoeft niet uit te leggen hoe de keuze wordt gemaakt).