Metody Programowania: Lista 8

Due on Monday, April 26, 2015

TWI 17.00-19.00

Bartosz Bednarczyk

Spis treści

Metody programowania 2015

Lista zadań nr 8

Na zajęcia 27–30 kwietnia 2015

Zadanie 1 (1 pkt). Składnię abstrakcyjną drzew binarnych bez etykiet będziemy reprezentować w postaci struktur zbudowanych z atomu `leaf/0` i binarnego funktora `node/2`. Składnię konkretną takich drzew opisuje gramatyka

$$\langle\text{tree}\rangle \;\to\; \texttt{*}$$
$$\langle\text{tree}\rangle \;\to\; \texttt{(}\ \langle\text{tree}\rangle\ \langle\text{tree}\rangle\ \texttt{)}$$

gdzie `*` oznacza liść, a para drzew ujęta w nawiasy oznacza drzewo, w którym synami korzenia są podane drzewa. Napisz gramatykę DCG odpowiadającą powyższej gramatyce. Czy generuje ona wszystkie słowa należące do języka opisanego tą gramatyką? Przepisz gramatykę DCG tak, by otrzymany program prologowy działał deterministycznie podczas rozpoznawania słów (taki program nie nadaje się oczywiście do generowania słów). Dodaj odpowiednie akcje semantyczne budujące abstrakcyjne drzewa rozbioru.

Zadanie 2 (1 pkt). Oto gramatyka zawierająca binarny operator łączący w lewo:

$$\langle\text{expression}\rangle \to \langle\text{simple expression}\rangle$$
$$\langle\text{expression}\rangle \to \langle\text{expression}\rangle\ \texttt{*}\ \langle\text{simple expression}\rangle$$
$$\langle\text{simple expression}\rangle \to \texttt{a}$$
$$\langle\text{simple expression}\rangle \to \texttt{b}$$
$$\langle\text{simple expression}\rangle \to \texttt{(}\ \langle\text{expression}\rangle\ \texttt{)}$$

Napisz odpowiednią gramatykę DCG odpowiadającą powyższej gramatyce. Czy generuje ona wszystkie słowa należące do języka opisanego tą gramatyką? Przepisz gramatykę DCG tak, by otrzymany program prologowy działał deterministycznie podczas rozpoznawania słów (taki program nie nadaje się oczywiście do generowania słów). Dodaj odpowiednie akcje semantyczne budujące abstrakcyjne drzewa rozbioru takich wyrażeń, zbudowane z atomów `a/0` i `b/0` oraz binarnego funktora `*/2`.

Zadanie 3 (1 pkt). Wiedząc, że

```
(f . g) x = f (g x)
flip f a b = f b a
curry f a b = f (a,b)
```

zdefiniuj w Haskellu funkcję `swap` spełniającą następującą specyfikację:

$$\texttt{flip}\,(\texttt{curry}\,f) \;=\; \texttt{curry}\,(f\ .\ \texttt{swap})$$

dla każdego $f :: (a, b) \to c$.

Zadanie 4 (1 pkt). Wiedząc, że

```
(f . g) x = f (g x)
f $ x = f x
flip f a b = f b a
head (x:_) = x
tail (_:xs) = xs
map _ [] = []
map f (x:xs) = f x : map f xs
```

wyznacz typy następujących wyrażeń w Haskellu:

```
(.)(.)
(.)($)
($)(.)
flip flip
(.)(.)(.)
(.)($)(.)
($)(.)($)
flip flip flip
tail $ map tail [[],['a']]
let x = x in x x
(λ _ → 'a') (head [])
(λ (_,_) → 'a') (head [])
```

Do wyznaczenia typów nie używaj kompilatora!

Zadanie 5 (1 pkt). Wiedząc, że

```
(f . g) x = f (g x)
fst (x,_) = x
snd (_,y) = y
pair (f,g) x = (f x, g x)
cross (f,g) = pair (f . fst, g . snd)
```

udowodnij, że:

$$\texttt{cross}\,(f,g)\ .\ \texttt{cross}\,(h,k) \;=\; \texttt{cross}\,(f\ .\ h,\ g\ .\ k).$$

Zadanie 6 (1 pkt). Mamy

```
class Enum a where
   fromEnum :: a → Int
   toEnum :: Int → a
```

Czy można zdefiniować typ (a, b) jako instancję klasy `Enum`, gdzie oba typy a i b są instancjami klasy `Enum`?

Zadanie 7 (1 pkt). Niech

```
data Nat = Zero | Succ Nat
(+), (*), (^) :: Nat → Nat → Nat
m + Zero = m
m + Succ n = Succ (m + n)
m * Zero = Zero
m * Succ n = (m * n) + m
m ^ Zero = Succ Zero
m ^ Succ n = (m ^ n) * m
```

Dla jakich wartości x, m, n :: `Nat` jest spełniona następująca równość:

$$x \mathbin{\hat{}} (m+n) \;=\; (x \mathbin{\hat{}} m) * (x \mathbin{\hat{}} n).$$

Zadanie 1

```
leaf.

tree(leaf) --> ['*'], !.

tree(node(A,B)) -->
        ['('],
        tree(A),
        tree(B),
        [')'].

% Przyklad uzycia :

?- phrase(tree(node(node(leaf, leaf), leaf)), ['(', '(', *, *, ')', *, ')']).
true.
```

Zadanie 1 continued on next page...

Zadanie 2

```
simp(a) --> "a",!.
simp(b) --> "b",!.
simp(E) --> "(",expr(E),")".

expr(E) --> simp(S),"*",!,expr1(E,S).
expr(S) --> simp(S).

expr1(E,A) --> simp(S),"*",!,expr1(E,A*S).
expr1(A*S,A) --> simp(S).

%expr(S) --> simp(S).
%expr([E,S]) --> expr(E),"*",simp(S).

:-
        expr(X,"(a*(a*b))*a",[]),
        print(X), print('\n'),
        expr(Y,"a*(a*b)*a",[]),
        print(Y), print('\n'),
        expr(Z,"a*((a*b)*a)",[]),
        print(Z).
```

Zadanie 2 continued on next page...

Zadanie 3

```
(f . g) x = f (g x)
(.) :: ((b->c)->(a->b)) -> (a->c)

flip f a b = f b a
flip :: (b -> a -> c) -> (a -> b -> c)

curry f a b = f(a, b)
curry :: ( (a,b)->c  ) -> (a->b->c)

uncurry f(a,b) = f a b
uncurry :: (a->b->c) -> (a,b) -> c

Zad. Zdefinuj swap tak aby flip(curry f) = curry(f . swap)
bylo prawda dla kazdej f :: (a,b) -> c.

Rozw :

Wezmy dowolne f :: (a,b) -> c
curry f :: a->b->c
flip(curry f) :: b->a->c  = typ curry(f . swap)

Zauwazmy, ze curry . uncurry = id = uncurry . curry
Zatem (f . swap) :: (b,a) -> c.

Stad prosty wniosek, ze swap :: (a,b) -> (b,a).
```

Zadanie 3 continued on next page. . .

Zadanie 4

```haskell
(f . g) x = f (g x)
f $ x = f x
flip f a b = f b a

(.) :: (b -> c) -> (a -> b) -> a -> c
($) :: (a -> b) -> a -> b
flip :: (a -> b -> c) -> b -> a -> c

g1 :: (a1 -> b -> c) -> a1 -> (a -> b) -> a -> c
g1 = (.) (.)

g2 :: (a1 -> a -> b) -> a1 -> a -> b
g2 = (.) ($)

g3 :: (b -> c) -> (a -> b) -> a -> c
g3 = ($) (.)

g4 :: b -> (a -> b -> c) -> a -> c
g4 = flip flip

g5 :: (b -> c) -> (a -> a1 -> b) -> a -> a1 -> c
g5 = (.) (.) (.)

g6 :: (b -> c) -> (a -> b) -> a -> c
g6 = (.) ($) (.)

g7 :: (a -> a1 -> b) -> a -> a1 -> b
g7 = ($) (.) ($)

g8 :: (a -> ((a1 -> b -> c1) -> b -> a1 -> c1) -> c) -> a -> c
g8 = flip flip flip

g9 :: [[Char]]
g9 = tail $ map tail [[], ['a']]

g10 :: t
g10 = let x = x in x x

g11 :: Char
g11 = (\_ -> 'a') (head [])

g12 :: Char
g12 = (\(_,_) -> 'a') (head [])
```

Zadanie 4 continued on next page. . .

Zadanie 5

```
Wiedzac, ze :

(1) : (f . g) x = f (g x)
(2) : fst (x,_) = x
(3) : snd (_, y) = y
(4) : pair (f, g) x = (f x, g x)
(5) : cross (f, g) = pair(f . fst, g. snd )

Udowodnij, ze cross(f, g) . cross(h, k) = cross( f . h, g . k )
```

Twierdzenie 1.

$$cross(f,g)\ .\ cross(h,k) = cross(f.h, g.k)$$

Dowód: Weźmy dowolne funkcje f, g, h, k i dowolny x - tak aby typy się zgadzały.

$$\begin{aligned}
(cross(f,g)\ .\ cross(h,k))\ (x) &\overset{1}{=} \\
\overset{1}{=} cross(f,g)(cross(h,k)(x)) &\overset{5}{=} \\
\overset{5}{=} cross(f,g)(pair(h\ .\ fst, k\ .\ snd)(x)) &\overset{4}{=} \\
\overset{4}{=} cross(f,g)((h\ .\ fst)(x), (k\ .\ snd)(x)) &\overset{5}{=} \\
\overset{5}{=} pair(f\ .\ fst, g\ .\ snd)((h\ .\ fst)(x), (k\ .\ snd)(x))) &\overset{4}{=} \\
\overset{4}{=} (f\ .\ fst((h.fst(x)), (k.snd)(x))), g\ .\ ((h.fst(x)), (k.snd)(x)))) &\overset{2,3}{=} \\
\overset{2,3}{=} ((f\ .\ (h\ .\ fst)(x)), (g\ .\ (k\ .\ snd)(x))) &\overset{4}{=} \\
\overset{4}{=} pair(f\ .\ h\ .\ fst, g\ .\ k\ .\ snd) &\overset{5}{=} \\
\overset{5}{=} cross(f\ .\ h, g\ .\ k)(x)
\end{aligned}$$

□

Zadanie 6

```
class Enum a where
        fromEnum :: a -> Int
        toEnum :: Int -> a

% Zad :
% Czy mozna zdefiniowac typ (a,b) instancji klasy enum,
% gdzie oba typy a i b sa instancjami klasy Enum.

% Odp:
% Tak, mozna.
% Przykladowa konstrukcja :

% fromEnum :: (a,b) -> Int
% fromEnum a b = fromEnum a + fromEnum b

% toEnum :: Int -> (a,b)
% toEnum i = (toEnum(i), toEnum(i))
```

Zadanie 6 continued on next page...

Zadanie 7

Niech

```
data Nat = Zero | Succ Nat

(+), (*), (^) :: Nat -> Nat -> Nat

m + Zero = m
m + Succ n = Succ (n + m)

m * Zero = Zero
m * Succ n = (m * n) + m

m ^ Zero = Succ Zero
m ^ Succ n = (m ^ n) * m
```

Twierdzenie 2.

$$\forall n, m, x \in Nat \ : \ x \uparrow (n + m) = (x \uparrow m) * (x \uparrow n)$$

Dowód: Weźmy dowolne $x, m \in Nat$. Będziemy dowodzić tego twierdzenia indukcyjnie względem n.

- $n = \perp$ - do uzupełnienia!
- $n = Zero$

$$L = x \uparrow (m + Zero) = x \uparrow m$$

$$P = (x \uparrow m)*(x \uparrow Zero) = (x \uparrow m)*Succ\,Zero = (x \uparrow m)+(x \uparrow m)*Zero = (x \uparrow m)+Zero = x \uparrow m = L$$

- Weźmy dowolne $n \in Nat$ i załóżmy dla niego prawdziwość tezy. Pokażmy, że zachodzi też dla $Succ\ n$.

$$L = x \uparrow (m+Succ\,n) = x \uparrow Succ\,(m+n) = (x \uparrow (m+n))*x \stackrel{zal}{=} (x \uparrow m)*(x \uparrow n)*x = (x \uparrow m)*(x \uparrow Succ\,n) = P,$$

co na mocy twierdzenia o indukcji daje tezę.

□

Uwaga 1. *W dowodzie skorzystaliśmy niejawnie z łączności operatora $*$. Musimy jeszcze pokazać, że w rzeczywistości mnożenie liczb jest łączne tj. $\forall x, y, z \in Nat \ : \ x * (y * z) = (x * y) * z$. Dowód można znaleźć tutaj: Proof wiki.*