Programowanie Funkcyjne
WPPT
Lista zadań

Jacek Cichoń, WPPT, PWr, 2018/19

Zadania oznaczone * są nieco trudniejsze od zadań bez gwiazdki. Zadania oznaczone ** są jeszcze trudniejsze.

1 Wstęp

1.1 Wprowadzenie do Haskell'a

Zadanie 1 — Zrób wszystkie zadania z książki Real World Haskell po rozdziale pierwszym.

Zadanie 2 — Oblicz w GHCI wartości wyrażeń $2 \wedge 3 \wedge 2$, $(2 \wedge 3) \wedge 2$ i $2 \wedge (2 \wedge 3)$. Dowiedz się jaka jest łączność operatora $\wedge$ za pomocą polecenia :i $(\wedge)$.

Zadanie 3 — Funkcją Eulera ϕ nazywamy funkcję określoną wzorem

$$\phi(n) = \text{card}(\{k \leq n : \text{gcd}(k, n) = 1\}) \, .$$

o dziedzinie $\mathbb{N}^+$.

1. Oprogramuj funkcję ϕ (funkcja gcd jest w bibliotece Prelude)
2. Napisz funkcję, która dla danej liczby naturalnej n wyznacza liczbę $\sum_{k|n} \phi(k)$.

Zadanie 4 — Trójkę liczb naturalnych (a, b, c) nazywamy właściwą trójką pitagorejską jeśli $a^2 = b^2 + c^2$ oraz $\text{gcd}(b, c) = 1$. Wyznacz wszystkie właściwe trójki pitegorejskie takie, że $a \leq 200$.

Zadanie 5 — Zaimplementuj na kilka sposobów funkcję służącą do wyznaczania liczb Fibbonacciego: rekurencyjnie, rekurencyjnie za pomocą wzorców.

Zadanie 6 — Zaimplementuj funkcję $\binom{n}{k}$. Nie stosuj tożsamości $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ - jest to kosztowne rozwiązanie; zastosuj wzór rekurencyjny na $\binom{n+1}{k+1}$.

Zadanie 7 — Liczbę naturalną n nazywamy doskonałą jeśli $n = \sum\{d < n : d|n\}$. Np. $6 = 1 + 2 + 3$. Wyznacz wszystkie liczby doskonałe mniejsze od 10000.

1.2 Elementy teorii kategorii

Zadanie 8 — Które z następujących struktur są monoidami?

1. $([0, 1], 0, \vee)$, gdzie $x \vee y = \max\{x, y\}$
2. $([0, 1], 1, \wedge)$, gdzie $x \wedge y = \min\{x, y\}$
3. $((0, \infty), 1, \star)$, gdzie $x \star y = x^y$
4. $(X^X, \text{Id}_x, \circ)$ (X jest ustalonym zbiorem)
5. $(X^*, [], ++)$, (gdzie X jest ustalonym zbiorem)

Zadanie 9 — Pokaż, że strzałka Id_A jest jednoznaczna, czyli, że jeśli $\text{Id}1_A$ oraz $\text{Id}2_A$ spełniają własności identyczności to $\text{Id}1_A = \text{Id}2_A$.

Zadanie 10 — Pokaż, że złożenie monomorfizmów jest monomorfizmem.

Zadanie 11 — Pokaż, że złożenie epimorfizmów jest epimorfizmem. Spróbuj podać proste uzasadnienie tego faktu oparte o poprzednie zadanie.

Zadanie 12 — Pokaż, że jeśli $f : A \to B$ jest izomorfizmem, to odwrotność f^{-1} jest wyznaczona jednoznacznie.

Zadanie 13 — Pokaż, że jeśli f^{-1} jest odwrotnością $f : A \to B$ i g^{-1} jest odwrotnością $g : B \to C$, to $f^{-1} \circ g^{-1}$ jest odwrotnością $g \circ f : A \to C$.

Zadanie 14 — Podaj przykład kategorii ze strzałką która jest monomorfizmem oraz epimorfizmem, ale nie jest izomorfizmem.

Zadanie 15 — Rozważamy kategorię zbudowaną z częściowego porządku $(X, \leq)$. Kiedy istnieją w niej elementy początkowe i końcowe?

Zadanie 16 — Zinterpretuj w języku informatyki komutowanie następującego diagramu

$$\begin{array}{ccc} Int & \xrightarrow{succ_{Int}} & Int \\ {\scriptstyle toReal}\downarrow & & \downarrow{\scriptstyle toReal} \\ Real & \xrightarrow[succ_{Real}]{} & Real \end{array}$$

Zadanie 17 — Pokaż, że obiekty końcowe (terminalne) w ustalonej kategorii są wyrażone jednoznacznie z dokładnością do izomorfizmu. Pokaż podobną własność obiektów początkowych.

Zadanie 18 — Wyznacz obiekty końcowe i początkowe w następujących kategoriach:

1. w kategorii grup $\mathbf{Grp}$
2. w kategorii ciał
3. w kategorii częściowych porządków $\mathbf{Pos}$
4. w kategorii monoidów $\mathbf{Mon}$
5. $\mathbf{Set} \times \mathbf{Set}$
6. $\mathbf{Set}^{\to}$

Zadanie 19 — Pokaż, że produkt $(A \times B, \pi_A, \pi_B)$ jest wyznaczony jednoznacznie z dokładnością do izomorfizmu w kategorii $\mathbf{Set}$. Wskazówka: Skorzystaj z jednoznaczności mediatora w definicji produktu.

2 Podstawowe typy

2.1 Haskell

Zadanie 20 — Dowiedz się jak można przekonwertować elementy typu Int oraz Integer na typy Float i Double.

Zadanie 21 — Oszacuj złożoność obliczeniową następującej funkcji służącej do odwracania listy:

```
rev :: [a] -> [a]
rev []     = []
rev (x:xs) = (rev xs) ++[x]
```

Zadanie 22 — Oprogramuj funkcję fib, która wyznacza n-tą liczbę Fibonacciego w czasie liniowym. Wskazówka: przyglądnij się parze (f_{n+1}, f_n).

Zadanie 23 — Napisz funkcję middle, która dla trzech podanych liczb x, y, z wyznacza która z nich leży pomiędzy dwoma pozostałymi.

Zadanie 24 — Ulepsz implementację funkcji Quick Sort która omówiona była na wykładzie.

Zadanie 25 — Napisz funkcję inits, która dla danej listy wyznaczy listę wszystkich jej odcinków początkowych, np.

`inits [1,2,3,4] == [[],[1],[1,2],[1,2,3],[1,2,3,4]]`

Zadanie 26 — Napisz funkcję `partitions`, która dla danej listy xs wyznaczy liste wszystkich par (ys,zs) takich, że `xs == ys++zs`.

Zadanie 27 — Napisz funkcje nub, która usunie z listy wszystkie duplikaty, np. `nub [1,1,2,2,2,1,4,1] == [1,2,4]`

Zadanie 28 — Napisz funkcje `permutations`, ktora dla danej listy wyznaczy listę wszystkich jej permutacji (możemy założyć, ze wszystkie elementy listy wejściowej sa różne).

Zadanie 29 — Napisz funkcję, która oblicza iloma zerami (w układzie dziesiętnym) kończy się liczba $n!$. **Uwaga:** taki pomysł: „mam dane n; obliczam $n!$; zamieniam na łańcuch s; odwracam go; liczę ilość początkowych zer" traktujemy jako kompletnie beznadziejny.

2.2 Teoria Kategorii

Zadanie 30 — Pokaż, że jeśli $\mathcal{C}$ jest kategorią, to $\mathcal{C}^{op}$ też jest kategorią.

Zadanie 31 — Niech $(G, \cdot)$ będzie grupą. Na zbiorze G określamy działanie $a \star b = b \cdot a$

1. Pokaż, że $(G\star)$ jest grupą.
2. Znajdź izomorfizm między grupami $(G, \cdot)$ oraz $(G, \star)$.
3. Jaki ma związek to zadanie z konstrukcją $\mathcal{C}^{op}$?

Zadanie 32 — Zapisz w języku diagramów komutujących (przemiennych) łączność operacji złożenia funkcji.

Zadanie 33 — Niech $(G, \cdot)$, $(H, \star)$ będą grupami oraz niech $f : G \to H$ będzie homomorfizmem. Wiadomo, że grupa ilorazowa $G/\ker(f)$ jest izomorficzna z podgrupą $\text{img}(f)$ grupy H. Zapisz to w języku przemiennych diagramów.

Zadanie 34 — Niech $\mathcal{C}$ będzie kategorią z produktem. Dla $f : X \to A$ oraz $g : Y \to B$ niech $\langle f, g\rangle$ będzie strzałką mediacyjną dla produktu $A \times B$. Pokaż, że $\langle f \circ h, g \circ h\rangle = \langle f, g\rangle \circ h$.

Zadanie 35 — Rozważamy kategorię **Mon** (monoidów). Pokaż, że funkcja $f : (\mathbb{N}, 0, +) \to (\mathbb{Z}, +, +)$ określona wzorem $f(x) = x$ (czyli identyczność na $\mathbb{N}$ traktowana jako morfizm z $(\mathbb{N}, 0, +)$ do $(\mathbb{Z}, 0, +)$) jest epimorfizmem.

Zadanie 36 — Ustalmy zbiór Ω. Rozważmy kategorię $\mathcal{P}(\Omega)$ której obiektami są wszystkie podzbiory zbioru Ω, zaś morfizmy oznaczają zawieranie zbiorów. Niech $A, B \subseteq \Omega$.

1. Wyznacz $A \times B$ w tej kategorii
2. Wyznacz $A + B$ w tej kategorii.

Zadanie 37 — Rozważmy takie przyporządkowanie $F : \mathbf{Set} \to \mathbf{Set}$

$$F(X) = \begin{cases} \emptyset & : |X| < \aleph_0 \\ \mathbb{N} & : |X| \geq \aleph_0 \end{cases}$$

Pokaż, że F nie można rozszerzyć do funktora.

Zadanie 38 — Sprawdź, że następujące przyporządkowania są endofunktorami kategorii **Set** oraz zaproponuj dla nich odwzorowania $\eta_X : X \to F(X)$ i $\mu_X : F(F(X)) \to F(X)$:

1. $F(X) = X \times X$; $F(f : X \to Y)(x, y) = (f(x), f(y)$
2. (Reader) $R_A(X) = X^A$; $R(f : X \to Y)(\phi) = f \circ \phi$
3. (Writer) $W_{\mathcal{M}}(X) = M \times X$; $W_{\mathcal{M}}(f : X \to Y) = id_M \times f$, gdzie $\mathcal{M} = (M, e, \star)$ jest ustalonym monoidem
4. (State) $S_A(X) = (A \times X)^A$; $S_A(f : X \to Y)(\phi) = (\lambda a)(\text{let } (b, y) = \phi(a) \text{ in } (b, f(y)))$
5. (Maybe) $M(X) = X \cup \{\uparrow_X\}$; $F(f : X \to Y) = f \cup \{(\uparrow_X, \uparrow_Y)\}$
6. (Niedeterminizm) $N(X) = \{Y \subseteq X : |Y| < \aleph_0\}$; $N(f : X \to Y)(A) = f[A]$

Odwzorowania η oraz μ muszą posiadać te same własności co ich odpowiedniki dla funktora `List`, czyli $\mu_X \circ \eta_{F(X)} = id_{F(X)}$, $\mu_X \circ F(\eta_X) = id_{F(X)}$ oraz $\mu_X \circ \mu_{F(X)} = \mu \circ F(\mu)$.

Zadanie 39 — Niech `mmap f = map (map f)` oraz `mmmap f = map (map (map f))`.

1. Zbadaj typy tych odwzorowań.
2. Przetestuj ich działanie
3. Pokaż, że `mmap = map . map` oraz `mmmap = map . map . map`

3 Lambda wyrażenia

Zadanie 40 — Zapisz operacje binarne $(+)$, $(*)$ za pomocą lambda wyrażeń.

Zadanie 41 — Niech `ff = (2 ^)` oraz `gg = (^ 2)`. Podaj interpretacją tych funkcji.

Zadanie 42 — Sprawdź wartości wyrażeń
`map (^ 2) [1..5]`
oraz
`map (2 ^) [1..5]`
i wyjaśnij dokładnie otrzymane wyniki.

Zadanie 43 — Niech `f x y = (\x -> (\y -> x + 2 * y)) (x * y)`. Zastosuj alfa transformację i beta redukcję do uproszczenia funkcji `f`.

Zadanie 44 — Rozważmy następujące definicje
```
f = \x -> x * x
g = \y -> f (f y)
h = g . g
```
Uprość funkcję `h`.

Zadanie 45 — Spróbuj uprościć następujące wyrażanie: $(\lambda x \to (x\ x))(\lambda x \to x)$. Wskazówka: Zastosuj najpierw alfa transformację do pierwszego składnika.

Zadanie 46 — Oblicz w GHCi następujące wyrażenie
`(head $ map (\x y -> (x * x) + (y * y) ) [2 ,3 ,4]) 5`
i wyjaśnij otrzymany wynik.

Zadanie 47 — (Sanity test) Niech

1. $S = (\lambda f\ g\ x \to f\ x\ (g\ x))$
2. $K = (\lambda x\ y \to x)$
3. $I = (\lambda x \to x)$

Pokaż, że $S\ K\ K = I$

4 Listy

Zadanie 48 — Funkcja `span` o sygnaturze `span :: (a -> Bool) -> [a] -> ([a], [a])` zastosowana do predykatu p i listy xs, zwraca krotkę, w której pierwszy element jest najdłuższym prefiksem (być może pustym) xs elementów spełniających p, a drugi element jest pozostałą częścią listy. Sprawdź jej działanie w GHCI i następnie samodzielnie ją zaprogramuj. Wskazówka: Skorzystaj z wyrażenia let.

Zadanie 49 — Napisz funkcję która eliminuje kolejne duplikaty z listy, np. `ecd [1,1,1,2,2,1,1,1]` $\Rightarrow$ `[1,2,1]`.

Zadanie 50 — Napisz funkcję która pakuje kolejne duplikaty do podlisty, np. `pack [1,1,1,2,2,3,3,3]` $\Rightarrow$ `[[1,1,1],[2,2],[3,3,3]]`. Wskazówka: Pomyśl o pomocniczej funkcji trzech zmiennych `f current ys acc`; w liście list `acc` możesz zbierać podciągi; `current` jest aktualnie budowanym podciągiem, zaś `ys` jest przetwarzanym ciągiem.

Zadanie 51 — Kodowanie RLE (Run-Length Encoding) polega na zapisaniu ciągów tych liter za pomocą liczników powtórzeń. np. `rleEncode "aaaabbbaaaaccd"` ⇒ `[(4,'a'),(3,'b')(4,'a'),(2,'c'),(1,d)]`.

1. Zaimplementuj tę funkcję
2. Napisz funkcję odwrotną `rleDecode` do funkcji `rleEncode`.

5 Folds

Zadanie 52 — Zdefiniuj za pomocą funkcji `foldr` funkcję, które dla listy liczb $[a_1, \ldots, a_n]$ oblicza ile liczb parzystych występuje w tej liście.

Zadanie 53 — Sprawdź typy i przetestuj działanie funkcji `sum`, `product`, `all` i `any`.

Zadanie 54 — Przetestuj działanie funkcji `foldl (+) 0 X`, `foldr (+) 0 X`, `foldl' (+) 0 X`, `foldr' (+) 0 X` oraz `sum X` na dużych listach liczb `X`. Wskazówka: skorzystaj z polecenia GHCi `:set +s`; w celu usunięcia wyświeltania informacji skorzystaj z polecenia `:unset +s`.

Zadanie 55 — Korzystając z funkcji `foldl` i `foldr` napisz funkcję approx n zdefiniowaną następująco

$$\text{approx}(n) = \sum_{k=1}^{n} \frac{1}{k!}$$

Zadanie 56 — Napisz, korzystając z funkcji `foldl`, funkcję która dla ciągu liczb $[a_1, \ldots, a_n]$ oblicza $\sum_{k=1}^{n} (-1)^{k+1} a_k$

Zadanie 57 — Napisz funkcję która dla zadanej listy $[a_1, \ldots, a_n]$ elementu typu [Fractional a] wyznaczy średnią arytmetyczną oraz wariancję ciągu $(a_1, \ldots, a_n)$. Skorzystaj tylko raz z funkcji `fold`.

6 Naturalne transformacje

Zadanie 58 — Niech $F : \mathbf{Set} \to \mathbf{Set}$ będzie przyporządkowaniem określonym wzorami $F(X) = \mathbb{N} \times X$ i $F(f : X \to Y)(n, x) = (n + 1, fx)$. Pokaż, że F nie jest funktorem.

Zadanie 59 — Niech $S(X) = X \times X$ i $S(f : X \to Y)((x, y)) = (f(x), f(y))$.

1. Wyznacz wszystkie naturalne transformacje $\eta : S \dot{\to} S$.
2. Wyznacz strukturę monoidu $(\text{Nat}(S, S), \circ)$

Zadanie 60 — Niech $F(X) = P(X)$ oraz $F(f : X \to Y) = (\lambda A \to f[A])$.

1. Wyznacz $Nat(I, F)$
2. Wyznacz $Nat(F, I)$.

Zadanie 61 — Ustalmy zbiór A i rozważmy funktory $F, G : \mathbf{Set} \to \mathbf{Set}$ określone wzorami $F(X) = A \times X$, $G(X) = X \times A$, oraz $F(f)(a, x) = (a, f(x))$ i $G(f)(x, a) = (f(x), a)$ dla $f : X \to Y$ oraz $a \in A$ i $x \in X$. Wyznacz wszystkie naturalne transformacje oraz wszystkie naturalne izomorfizmy $\eta : F \dot{\to} G$.

Zadanie 62 — Rozważamy funktor zbioru potęgowego $P : \mathbf{Set} \to \mathbf{Set}$ ze działaniem na strzałkach w $\mathbf{Set}$ określonym wzorem $P(f)(A) = \vec{f}(A)$ dla $f : X \to Y$ (gdzie $\vec{f}(A)$ oznacza obraz zbiory A przez odwzorowanie f). Wyznacz wszystkie naturalne transformacje $\eta : I \dot{\to} P$, gdzie I jest funktorem identycznościowym (czyli $I(X) = X$ i $I(f) = f$).

7 Typy

Zadanie 63 — Niech $m91$: Int $\rightarrow$ Int będzie funkcją zdefiniowaną wzorem

$$m91(x) = \begin{cases} n - 10 & : n > 100 \\ m91(m91(n + 11)) & : n <= 100 \end{cases}$$

Zaimplementuj funkcję $m91$ w języku Haskell i wyznacz jej wartości dla wszystkich liczb ze zbioru $\{0, \ldots, 100\}$.

Zadanie 64 — Zdefiniuj typ IntOrString (koprodukt typów `Int` i `String`) i zaimplementuj jego instację do klasy Eq.

Zadanie 65 — Rozważmy typ

```
data BTree a = L a | N a (BTree a) (BTree a) deriving Eq
```

1. Zrób z typu BTree własną instancję klasy Show
2. Zrób z typu BTree instancję klasy Functor
3. Pokaż, że zaimplementowana funkcja fmap spełnia aksjomaty funktora. Sprawdź na kilku przykładach, że twoja implementacja działa poprawnie.
4. Zrób z typu BTree instancję klasy Foldable - czyli zaimplementuj funkcję foldr
5. Napisz za pomocą funkcji foldr funkcję findInBTree, o sygnaturze

   ```
   findInBTree :: a -> BTree a -> Bool
   ```

   która sprawdza, czy w drzewie występuje węzeł z daną jako pierwszy argument wartością
6. Napisz za pomocą funkcji foldr funkcję, która zlicza ilość węzłów w drzewie
7. Napisz funkcję concatBTree o sygnaturze

   ```
   concatBTree :: a -> BTree a -> BTree a -> BTree a
   ```

   która z pierwszego parametru i dwóch drzew robi jedno drzewo.
8. Napisz wariant funkcji fold dla BTree o następującej sygnaturze

   ```
   foldTree :: (z->a->z->z) -> z-> BTree a -> z
   ```

   która na liściach zachowuje się następująco: `foldTree f z (L a) = z`. Zdefiniuj za pomocą tej funkcji funkcje które wyznaczają wysokość drzewa i liczbę liści w drzewie

Zadanie 66 — Dla liczby naturalnej n definiujemy

1. $V_n(X) = \underbrace{X \times \ldots \times X}_{n}$

2.$V_n(f : X \rightarrow Y)((x_1, \ldots, x_n)) = (fx_1, \ldots, fx_n)$

1. Pokaż, że V_n jest funktorem.
2. Wyznacz liczbę naturalnych transformacji w V_n w V_m
3. Wyznacz wszystkie elementy $Nat(V_2, V_3)$

Zadanie 67 — Niech f, g : Float $\rightarrow$ Maybe Float.

1. Zaimplementuj samodzielnie $mx >>= f$
2. Zaimplementuj samodzielnie $f >=> g$
3. Zaimplementuj samodzielnie funkcję `join` dla funktora `Maybe`.

Zadanie 68 — Rozważamy funktory z zadania 38

1. Sprawdź, czy wszystkie warianty odwzorowań η i μ są naturalnymi transformacjami. Jeśli nie, to popraw zaproponowane definicje.
2. Sprawdź, czy dla wszystkich par odwzorowań η i μ spełnione są równości $\mu_X \circ \eta_{F(X)} = id_{F(X)}$, $\mu_X \circ F(\eta) = id_{F(X)}$ oraz $\mu_X \circ \mu_{F(X)} = \mu_X \circ F(\mu_X)$. Jeśli nie, to popraw definicje.

Zadanie 69 — Zostaw telefon, komputer, książki i notatki w domu. Zabarykaduj się w jakimś pokoju na uczelni na tak długo, aż samodzielnie udowodnisz lemat Yonedy.

Zadanie 70 — Wyjaśnij następującą równoważność:

kategorie z jednym obiektem $\equiv$ monoidy

Zadanie 71 — Wyznacz elementy początkowe i końcowe w kategorii endomorfizmów [Set,Set].

Zadanie 72 — Zadanie 73 — Niech M będzie ustalonym niepustym zbiorem. Niech $W(X) = X \times M$ i $W(f : X \to Y) = f \times id$. Załóżmy, że trójka (W, η, μ) jest monadą. Niech $e \in M$ będzie takie, że $\eta_{\{1\}} = (1, e)$. Niech $\star : M \times M \to M$ będzie takie, że $\mu(((1, n), m)) = (1, n \star m)$. Pokaż, że struktura $(M, e, \star)$ jest monoidem.

Zadanie 74 — Załóżmy, że $\mathcal{M} = (M, m, \oplus)$ i $\mathcal{N} = (N, n, *)$ są monoidami. Na zbiorze $M \times N$ określamy działanie $(x, a) \otimes (y, b) = (x \oplus y, a * b)$. Pokaż, że struktura $\mathcal{M} \times \mathcal{N} = (M \times N, (m, n), \otimes)$ jest monoidem.

Zadanie 75 — Niech Sum Int $= (\mathbb{Z}, 0, +)$, Product Int $= (\mathbb{Z}, 1, \cdot)$.

1. Czy monoidy Sum Int oraz Product Int są izomorficzne?
2. Czy monoidy Sum Int oraz (Sum Int) $\times$ (Sum Int) są izomorficzne?

8 Monady

Zadanie 76 — Rozważmy funktor $V_3(X) = X \times X \times X, V_3(f : X \to Y)((x_1, x_2, x_3)) = (f(x_1), f(x_2), f(x_3))$. W Zadaniu 66 pokazaliśmy, że to jest rzeczywiście funktor.

1. Pokaż, że jedynym odwzorowaniem naturalnym $\eta \Rightarrow V_3$ jest odwzorowanie $\eta_X(x) = (x, x, x)$.
2. Pokaż, że jedynym odwzorowaniem $\mu : (V_3)^2 \Rightarrow V_3$ spełniającym równości $\mu \circ V_3(\eta_X) = id_{T(X)}$ i $\mu \circ \eta_{T(X)} = id_{T(X)}$ jest odwzorowanie zadane wzorem

$$\mu_X(((x_1, x_2, x_3), (y_1, y_2, y_3), (z_1, z_2, z_3))) = (x_1, y_2, z_3)\ .$$

3. Sprawdź, że trójka (V_3, η, μ) jest monadą.

Zadanie 77 — Rozważmy funktor V_3 z Zadania 76.

1. Zaimplementuj ten funktor jako konstruktor typów i zrób z niego instancję klasy funktor.
2. Wiedząc, że trójka (V_3, η, μ) jest monadą wyznacz dla niej operację $vx >>= f$
3. Zaimplementuj go jako monadę. Zastosuj najpierw kanoniczną metodę zbudowania z monady funktora aplikatywnego.
4. Oprogramuj samodzielnie operację <$*$>. Postaraj się aby zrobić to możliwie prosto. Zastąp poprzedni kod instancji Applicative ulepszonym kodem.

Zadanie 78 — Wyznacz dla wszystkich poznanych do tej pory monad operacje >>=, >> oraz <$*$>.

Zadanie 79 — Rozważmy następujący typ

```
data Term = V Int| Plus Term Term | Mult Term Term| Div Term
```

Oprogramuj `ewaluator eval:: Term -> Maybe Int` w stylu monadycznym.

Zadanie 80 — Pokaż, że następujące kody

```
do f<-tf; x<-tx; return (f x)
```

oraz

```
do f<- tf; fmap f tx
```

są równoważne.

Zadanie 81 — Zaimplementuj funkcję `myGCD :: Int -> Int -> Writer [String] Int` obliczającą największy wspólny dzielnik podanych dwóch liczb oraz wyznaczającą jednocześnie ślad wykonanych obliczeń. Po wywołaniu jej dla parametrów 32 i 14 powinniśmy otrzymać mniej więcej taki wynik:

```
(2,["32 mod 14 = 4","14 mod 4 = 2","4 mod 2 = 0","GCD = 2"])
```

Zadanie 82 — (Problem Collatz'a) Rozważmy funkcję `f::Int->Int` zdefiniowaną wzorem

```
f x = if x `mod` 2 ==0 then x `div` 2 else 3*x+1
```

Niech `h(x) = if x==1 then 1 else f x`. Interesuje nas ile iteracji funkcji `f` wewnątrz funkcji `g` jest potrzebnych do osiągnięcia 1. Hipoteza Collatz'a głowi, że funkcja `h` jest całkowita. Skorzystaj z monady `Writer (Sum Int,String) Int` do sprawdzania tej hipotezy. Składnik `Sum Int` to monoid liczb całkowitych z operacją `+`. Ma on służyć do zliczania wywołań funkcji `f`. Składnik `String` ma służyć do zapamiętania śladu obliczeń. Sprawdź, czy dla liczby 97 funkcja `h` wykonuje 118 kroków.

8.1 Notacja do

* **Zadanie 83 —** Niech data `Tree a = L a | B (Tree a) (Tree a)`.

1. Zrób z konstruktora `Tree` instancję klasy `Functor`
2. Zrób z konstruktora `Tree` instancję klasy `Monad`. Jego instancję klasy `Applicative` wykonaj za pomocą kanonicznej konstrukcji

   ```
   fT <*> xT = do f <- fT; x <- xT; return (f x)
   ```

3. Wyznacz funkcję `join` dla konstruktora `Tree`.
4. Zbadaj na prostych przykładach działanie operacji `<*>` i następnie spróbuj samodzielnie ją zaprogramować.

Zadanie 84 — Uprość wyrażenie `do x <- tx; f x` .

Zadanie 85 — Pokaż, że następujące kody:

```
do f <- tf; x <- tx; return (f x)
```

oraz

```
do f <- Tf; fmap f tx
```

są równoważne

Zadanie 86 — Oprogramuj funkcję eval:: `M.Map String Float -> Term -> Maybe Float` dla "języka"

```
data Term =   X String      -- zmienna
            | N Float       -- stała
            | E Term        -- exp
            | L Term        -- ln
            | A Term Term   -- add
            | S Term Term   -- substract
            | M Term Term   -- multiply
            | D Term Term   -- divide
```

Pierwszy parametr tej funkcji jest mapą przechowującą wartości zmiennych

1. Zrób infixową instancję klasy Show dla wyrażeń typu `Term`
2. Przetestuj działanie funkcji eval na pierwszej zmiennej w przypadkach, gdy nie wszystkie użyte w wyrażeniach zmienne mają przypisane wartości.

Zadanie 87 — Rozszyfruj działanie kodu `liftM (:[]) "abc"` .

8.2 Przykłady monad

Zadanie 88 — Niech

```
f :: Monad m => m (m a) -> m a
f mmx = do
  mx <- mmx
  x  <- mx
  return x
```

Jaką funkcję monadyczną oprogramowuje funkcja f? Czy można oprogramować funkcję f prościej?

Zadanie 89 — Rozważamy funktor $\mathrm{V2}(X) = X \times X$, $V(f : X \to Y)(x_1, x_2) = (f(x_1), f(x_2))$.

1. Oprogramuj ten funktor jako `data Pair a = P a a`.
2. Zrób z niego instancję klasy `Applicative` stosując jako `pure` jedyną naturalną transformację $\mathrm{I} \Rightarrow \mathrm{V2}$ oraz

```
(P f g) <*> (P x y) = P (f x) (g y)
```

3. Wiedząc, że trójka $(\mathrm{V2}, \eta, \mu)$, gdzie $\eta(x) = (x, x)$ oraz $\mu(((x_1, x_2), (y_1, y_2))) = (x_1, y_2)$ jest monadą zrób z V2 instancję klasy `Monad`
4. Sprawdź, że operacja `<*>` zdefiniowana w standardowy sposób, czyli jako

   ```
   pf <*> px = do {f<- pf; x<- px; return (f x)}
   ```

   pokrywa się z następującą implementacją

   ```
   (P f m) <*> (P x n) = P (f x) (m * n)
   ```

   gdzie * jest operacją monoidalną (jest to konieczny warunek zgodności monady z applicative)
5. Pokaż, że jeśli `f,g::a->b` oraz `px::Pair a`, to

   ```
   ( (P f g) <*> px ) = ( px >>= (\x -> P (f x) (g x)) )
   ```

Zadanie 90 — Oprogramuj za pomocą Monady Reader funkcję generującą stronę WWW służącą do wyświetlania „równania dnia". Zależeć ma od trzech parametrów typu string. Pierwszy parametr ma zawierać tytuł strony (title). Drugi ma być nazwą równania a trzeci parametr ma zawierać równanie zapisane w języku Latex. Fragment kodu może wyglądać mniej więcej tak:

```
html :: Reader (String,String,String) String
html = do
  t1<- head
  t2<- content
  return ( concat["<!DOCTYPE html>\n<html lang='pl'>\n",
                  t1, t2, "</html>"] )

head :: Reader (String,String,String) String
head = do .....
```

Zadanie 91 — Rozważamy siatkę (grid) $\mathbb{Z} \times \mathbb{Z}$ zaimplementowaną jako `type Grid = (Int,Int)`. Korzystając z monady typu `Writer String Grid` zaimplementuj funkcje `up,down,right,left :: Grid -> Writer String Grid` które oprogramowują skoki w górę, w dół, w lewo i w prawo o 1 tak aby Writer śledził sekwencje wywołań tych funkcji

1. Niech

   ```
   run pos = do
     x0 <- writer (pos,"")
     x1 <- up x0
     x2 <- left x1
     down x2
   ```

   Sprawdź, czy po wywołaniu `runWriter $ run (0,0)` otrzymasz `((-1,0),"uld")`.
2. Oprogramuj funkcję `contRun :: Tablica -> String -> Writer String Tablica`, której drugi parametr jest ciągiem sterującym: dla litery 'u' ma wywołać funkcję up, dla 'r' - funkcję right itd. Inne znaki powiiny być ignorowane. Po wywołaniu
   `runWriter $ contRun (0,0) "xxuuulllllddddddrraaa"`
   powinniśmy otrzymać odpowiedź
   `((-3,-3),"rrddddddlllllluuu").`

Zadanie 92 — Niech m będzie monadą. Niech

```
do
  x<- mx
  return (f x)
```

1. Jaka powinna być sygnatura funkcji f aby ten kod był poprawny semantycznie?
2. Zrób de-sugaryzację tego kodu.

9 Naturalne transformacje

Zadanie 93 — Niech $V2$ będzie funktorem z zadania 89. Niech $\pi^1((x_1, x_2)) = x_1$ oraz $\pi^2((x_1, x_2)) = x_2$. Są to naturalne transformacje z $V2$ do funktora identycznościowego I. Wyznacz $\pi^1 \star \pi^2$, $\pi^2 \star \pi^1$, $\pi^1 \star \pi^1$ oraz $\pi^2 \star \pi^2$.

Zadanie 94 — Niech $L(X) = [X]$, $\eta(x) = [x]$, $\mu(x) = \text{concat}(x)$. Wyznacz $\mu \star \eta$, $\eta \star \mu$, $\eta \star \eta$ oraz $\mu \star \mu$.

Zadanie 95 — Na wykładzie zdefiniowaliśmy złożenie horyzontalne naturalnych transformacji $\alpha : F \Rightarrow G$ oraz $\beta : K \Rightarrow L$ za pomocą wzoru

$$(\beta \star \alpha)_X = \beta_{G(X)} \circ K(\alpha_X)$$

1. Pokaż, że rzeczywiście $\beta \star \alpha$ jest naturalną transformacją z $K \circ F$ do $L \circ G$.
2. Pokaż, że $(\beta \star \alpha)_X = L(\alpha_X) \circ \beta_{F(X)}$. Wskazówka: Skorzystaj z tego, że β jest naturalną transformacją, zaś α jest morfizmem z $F(X)$ do $G(X)$.

Zadanie 96 — Niech $\mathcal{C}$, $\mathcal{D}$ oraz $\mathcal{E}$ będą kategoriami. Załóżmy, że $F_1, F_2, F_3 : \mathcal{C} \to \mathcal{D}$, $G_1, G_2, G_3 : \mathcal{D} \to \mathcal{E}$ są funktorami, oraz, że $\alpha_1 : F_1 \Rightarrow F_2$, $\alpha_2 : F_2 \Rightarrow F_3$, $\beta_1 : G_1 \Rightarrow G_2$, $\beta_2 : G_2 \Rightarrow G_3$ są naturalnymi transformacjami. Pokaż, że

$$(\beta_2 \circ \beta_1) \star (\alpha_2 \circ \alpha_1) = (\beta_2 \star \alpha_2) \circ (\beta_1 \star \alpha_1) \,.$$

C.D.N.

Powodzenia,
Jacek Cichoń