Testimisraamistik *QuickCheck*

Sissejuhatus

- *QuickCheck* on raamistik programmide testimiseks:
  - autorid: Koen Claessen ja John Hughes;
  - algselt ainult Haskellile, aga nüüd ka näiteks Erlangile.
- Võimaldab kontrollida kas programm rahuldab kirjeldatud omadusi mingil hulgal juhuslikult genereeritud sisenditel.
- Kasutamiseks:

```
import Test.QuickCheck
```

- Teegis on defineeritud kombinaatrid omaduste kirjeldamiseks, standartsete tüüpide jaoks juhuväärtuste generaatorid ning kombinaatorid juhuväärtuste generaatorite konstrueerimiseks kasutajadefineeritavatele andmetüüpidele.

QuickCheck

Näide: *reverse* omadusi

$$
\begin{aligned}
&prop\_RevRev &&:: [Int] \to Bool \\
&prop\_RevRev\ xs &&= reverse\ (reverse\ xs) == xs \\
&prop\_RevApp &&:: [Int] \to [Int] \to Bool \\
&prop\_RevApp\ xs\ ys &&= reverse\ (xs +\!\!+ ys) \\
& && \quad == reverse\ ys +\!\!+ reverse\ xs
\end{aligned}
$$

QuickCheck

Näide: *reverse* omadusi

$$
\begin{aligned}
&prop\_RevRev &&:: [Int] \to Bool \\
&prop\_RevRev\ xs &&= reverse\ (reverse\ xs) == xs \\
&prop\_RevApp &&:: [Int] \to [Int] \to Bool \\
&prop\_RevApp\ xs\ ys &&= reverse\ (xs +\!\!+ ys) \\
& && \quad == reverse\ ys +\!\!+ reverse\ xs
\end{aligned}
$$

Omaduste testimine

```
Main> quickCheck prop_RevRev
+++ OK, passed 100 tests.
Main> quickCheck prop_RevApp
+++ OK, passed 100 tests.
```

QuickCheck

Näide: mittekehtiv omadus

$$
\begin{aligned}
&prop\_RevWrong && :: [Int] \to [Int] \to Bool \\
&prop\_RevWrong\ xs\ ys = reverse\ (xs +\!\!+ ys) \\
&\qquad\qquad == reverse\ xs +\!\!+ reverse\ ys
\end{aligned}
$$

QuickCheck

Näide: mittekehtiv omadus

$$
\begin{aligned}
&prop\_RevWrong && :: [Int] \to [Int] \to Bool \\
&prop\_RevWrong\ xs\ ys && = reverse\ (xs +\!\!+ ys) \\
& && \quad == reverse\ xs +\!\!+ reverse\ ys
\end{aligned}
$$

Omaduse testimine

```
Main> quickCheck prop_RevWrong
*** Failed! Falsifiable (after 3 tests and 2 shrinks):
[0]
[1]
```

QuickCheck

Baasliides

$$quickCheck :: Testable\ prop \Rightarrow prop \rightarrow IO\ ()$$

- Funktsioon *quickCheck* saab argumendiks omaduse, mida testitakse juhuslikult genereeritud sisenditel.
- Vaikimisi testitakse 100 korda (kuid see on konfigureeritav).
- Kui mõni test ebaõnnestub, siis väljastatakse kontranäide (so. argumendid millel test ebaõnnestus).
  - Seejuures püütakse eelnevalt kontranäite suurust minimiseerida (*shrinking*).

Omadused

Omadused

```
class Testable prop where
      property :: prop → Property
instance Testable Bool
instance (Arbitrary a, Show a, Testable prop) ⇒
         Testable (a → prop)
```

- Omadused on suvalised avaldised, mille tüüp kuulub klassi *Testable*.
- Argumendid peavad reeglina olema monomorfset tüüpi.
  - Tarvilik selleks, et teada kuidas argumente genereerida.
- Nimekonventsioon: prefiks *prop_*
  - Pole kohustuslik!

Omadused

Näide: listide sorteerimine pistemeetodil

$$
\begin{aligned}
&isort :: Ord\ a \Rightarrow [a] \to [a] \\
&isort = foldr\ insert\ [\,] \\
&insert :: Ord\ a \Rightarrow a \to [a] \to [a] \\
&insert\ x\ [\,] \qquad\qquad\qquad\quad = [x] \\
&insert\ x\ (y : ys) \mid x \leqslant y \qquad = x : y : ys \\
&\qquad\qquad\qquad\quad \mid otherwise = y : insert\ x\ ys
\end{aligned}
$$

Omadused

Listide sorteerimine: Omadus 1

- Sorteeritud list peab olema järjestatud.

$$
\begin{aligned}
&prop\_sortOrder :: [Int] \to Bool \\
&prop\_sortOrder\ xs = ordered\ (isort\ xs) \\
&ordered :: Ord\ a \Rightarrow [a] \to Bool \\
&ordered\ (x : y : ys) \ = x \leqslant y \wedge ordered\ (y : ys) \\
&ordered\ ys \qquad\qquad = True
\end{aligned}
$$

Omadused

Listide sorteerimine: Omadus 2

- Sorteeritud listis on samad elemendid mis algses listis.

$$
\begin{aligned}
&prop\_sortElems :: [Int] \to Bool \\
&prop\_sortElems\ xs = sameElems\ xs\ (isort\ xs) \\
&sameElems :: Eq\ a \Rightarrow [a] \to [a] \to Bool \\
&sameElems\ xs\ ys \quad = null\ (xs \setminus\setminus ys) \wedge null\ (ys \setminus\setminus xs)
\end{aligned}
$$

Omadused

Testandmete inspekteerimine

$$collect :: (Show\ a, Testable\ prop) \Rightarrow a \rightarrow prop \rightarrow Property$$

- Funktsioon *collect* kogub infot genereeritavate testandemete kohta ning väljastab selle pärast testide sooritamist.

Omadused

Testandmete inspekteerimine

$$collect :: (Show\ a, Testable\ prop) \Rightarrow a \rightarrow prop \rightarrow Property$$

- Funktsioon *collect* kogub infot genereeritavate testandemete kohta ning väljastab selle pärast testide sooritamist.

Palju sisendeid oli mittetühi?

```
Main> let p = prop_sortOrder
Main> quickCheck (λxs → collect (null xs) (p xs))
+++ OK, passed 100 tests.
93% False
 7% True
```

Omadused

Testandmete inspekteerimine

$$collect :: (Show\ a, Testable\ prop) \Rightarrow a \rightarrow prop \rightarrow Property$$

- Funktsioon *collect* kogub infot genereeritavate testandemete kohta ning väljastab selle pärast testide sooritamist.

Milline oli sisendite pikkusi?

$$\texttt{Main>}\ quickCheck\ (\lambda xs \rightarrow collect\ (length\ xs\ `div`\ 20)\ (p\ xs))$$

```
+++ OK, passed 100 tests.
56% 0
23% 1
11% 2
 7% 3
 3% 4
```

Omadused

Testandmete inspekteerimine

$$collect :: (Show\ a, Testable\ prop) \Rightarrow a \rightarrow prop \rightarrow Property$$

- Funktsioon *collect* kogub infot genereeritavate testandemete kohta ning väljastab selle pärast testide sooritamist.

Mis olid tegelikud sisendid?

```
Main> quickCheck (λxs → collect xs (p xs))
+++ OK, passed 100 tests.
 1% [14,5,-10,4,-9,10,-4,-1,7]
 1% [1,-3,3,2,0,0]
 1% [0,0]
...
```

Omadused

Pistemeetodi omadus (ver. 1)

- Elemendi lisamisel sorteeritud listi jääb list sorteerituks.

$$
\begin{aligned}
&prop\_insertOrder1 :: Int \to [Int] \to Bool \\
&prop\_insertOrder1\ x\ xs \\
&\qquad = ordered\ xs\ `implies`\ ordered\ (insert\ x\ xs) \\
&implies :: Bool \to Bool \to Bool \\
&implies\ x\ y = not\ x \vee y
\end{aligned}
$$

Omadused

Pistemeetodi omadus (ver. 1)

- Elemendi lisamisel sorteeritud listi jääb list sorteerituks.

$$
\begin{aligned}
&prop\_insertOrder1 :: Int \to [Int] \to Bool \\
&prop\_insertOrder1\ x\ xs \\
&\qquad = ordered\ xs\ `implies`\ ordered\ (insert\ x\ xs) \\
&implies :: Bool \to Bool \to Bool \\
&implies\ x\ y = not\ x \vee y
\end{aligned}
$$

Probleem!

```
Main> let p = prop_insertOrder1
Main> quickCheck (λx xs → collect (ordered xs) (p x xs))
+++ OK, passed 100 tests.
89% False
11% True
```

Omadused

Tingimuslikud omadused

$(==>) :: Testable\ prop \Rightarrow Bool \rightarrow prop \rightarrow Property$

instance *Testable Property*

- Tüüp *Property* võimaldab kodeerida mitte ainult omaduse kehtimist, vaid ka ignoreerida teatavaid testandmeid.
- Kombinaator (==>) ignoreerib testmise sisendeid, millede korral eeldus ei ole täidetud ning genereerib nende asemel uued sisendid.
- Uuesti genereerimiste arv on vaikimisi 500 (on kofigureeritav).

Omadused

Pistemeetodi omadus (ver. 2)

- Elemendi lisamisel sorteeritud listi jääb list sorteerituks.

$$
\begin{aligned}
&prop\_insertOrder2 :: Int \to [Int] \to Property \\
&prop\_insertOrder2\ x\ xs = ordered\ xs \Longrightarrow \\
&\qquad\qquad\qquad\qquad\qquad\qquad ordered\ (insert\ x\ xs)
\end{aligned}
$$

Omadused

Pistemeetodi omadus (ver. 2)

- Elemendi lisamisel sorteeritud listi jääb list sorteerituks.

$$prop\_insertOrder2 :: Int \to [Int] \to Property$$
$$prop\_insertOrder2\ x\ xs = ordered\ xs \Longrightarrow ordered\ (insert\ x\ xs)$$

Probleem!

```
Main> let p = prop_insertOrder2
Main> quickCheck (λx xs → collect (ordered xs) (p x xs))
*** Gave up! Passed only 45 tests (100% True).
```

Omadused

Etteantud generaatoriga omadused

$$forAll :: (Show\ a, Testable\ prop) \Rightarrow \\ Gen\ a \rightarrow (a \rightarrow prop) \rightarrow Property$$

- Kombinaator *forAll* saab esimese argumendina generaatori, mida kasutatakse omadusele (so. teisele argumendile) antavate sisendite genereerimiseks.
- Võimaldab kasutada spetsiaalseid generaatoreid, mis garanteerivad sisenditel teatud omaduste kehtimise.

Omadused

Pistemeetodi omadus (ver. 3)

- Elemendi lisamisel sorteeritud listi jääb list sorteerituks.

$$
\begin{aligned}
&prop\_insertOrder3 :: Int \to Property \\
&prop\_insertOrder3\ x = forAll\ orderedList\ (\lambda xs \to \\
&\qquad\qquad\qquad\qquad ordered\ (insert\ x\ xs))
\end{aligned}
$$

Omadused

Pistemeetodi omadus (ver. 3)

- Elemendi lisamisel sorteeritud listi jääb list sorteerituks.

$$prop\_insertOrder3 :: Int \to Property$$
$$prop\_insertOrder3\ x = forAll\ orderedList\ (\lambda xs \to ordered\ (insert\ x\ xs))$$

Omaduse testimine

```
Main> quickCheck (forAll orderedList ordered)
+++ OK, passed 100 tests.
Main> quickCheck prop_insertOrder3
+++ OK, passed 100 tests.
```

Generaatorid

Generaatorid

$$\textbf{newtype}\ Gen\ a = \ldots$$
$$\textbf{instance}\ Monad\ Gen$$
$$\textbf{instance}\ Functor\ Gen$$
$$\textbf{instance}\ (Testable\ prop) \Rightarrow Testable\ (Gen\ prop)$$

- Generaatorid kuuluvad abstraksesse andmetüüpi Gen.
- Gen on monaad, mille efektiks on "ligipääs" juhuslikult genereeritud arvudele.

Genereeritavate näiteväärtuste väljastamine

$$sample :: Show\ a \Rightarrow Gen\ a \rightarrow IO\ ()$$

Generaatorid

Eeldefineeritud generaatorite kombinaatoreid

$$
\begin{aligned}
&\mathit{choose} && :: \mathit{Random}\ a \Rightarrow (a, a) \rightarrow \mathit{Gen}\ a \\
&\mathit{elements} && :: [a] \rightarrow \mathit{Gen}\ a \\
&\mathit{oneof} && :: [\mathit{Gen}\ a] \rightarrow \mathit{Gen}\ a \\
&\mathit{frequency} && :: [(\mathit{Int}, \mathit{Gen}\ a)] \rightarrow \mathit{Gen}\ a \\
&\mathit{sized} && :: (\mathit{Int} \rightarrow \mathit{Gen}\ a) \rightarrow \mathit{Gen}\ a \\
&\mathit{vectorOf} && :: \mathit{Int} \rightarrow \mathit{Gen}\ a \rightarrow \mathit{Gen}\ [a]
\end{aligned}
$$

Generaatorid

Vaikimisi generaatorid

```
class Arbitrary a where
  arbitrary :: Gen a
  shrink    :: a → [a]
  shrink _  = []
```

- Klassi *Arbitrary* kuuluvad tüübid, mille jaoks on defineeritud vaikimisi kasutatav generaator *arbitrary*.
- Lisaks on klassis defineeritud meetod *shrink*, mida kasutatakse väiksemate kontranäidete genereerimiseks:
  - *shrink* väljastab etteantud väärtusest struktuurselt väiksemate väärtuste listi;
  - kui leitakse kontranäide, siis testitakse omadust uuesti funktsiooni *shrink* poolt väljastatud väärtustel nii kaua, kuni väiksemat kontranäidet ei leita.

Generaatorid

Lihtsad generaatorid

```
instance Arbitrary Bool where
  arbitrary = choose (False, True)
instance (Arbitrary a, Arbitrary b) ⇒ Arbitrary (a, b) where
  arbitrary = liftM2 (,) arbitrary arbitrary
data Color = Red | Blue | Green
instance Arbitrary Color where
  arbitrary = elements [Red, Blue, Green]
```

Generaatorid

Lihtsad generaatorid

```
instance Arbitrary a => Arbitrary (Maybe a) where
  arbitrary = oneof [return Nothing
                    ,liftM Just arbitrary]
```

Generaatorid

Lihtsad generaatorid

$$\begin{aligned}&\textbf{instance } Arbitrary\ a \Rightarrow Arbitrary\ (Maybe\ a)\ \textbf{where}\\ &\quad arbitrary = oneof\ [return\ Nothing\\ &\qquad\qquad\qquad\qquad , liftM\ Just\ arbitrary]\end{aligned}$$

Probleem

Pooled genereeritavaest väärtustest on $Nothing$!!

Generaatorid

Lihtsad generaatorid

```
instance Arbitrary a ⇒ Arbitrary (Maybe a) where
  arbitrary = oneof [return Nothing
                    , liftM Just arbitrary]
```

Probleem

Pooled genereeritavaest väärtustest on *Nothing*!!

Erineva sagedusega generaatorid

```
instance Arbitrary a ⇒ Arbitrary (Maybe a) where
  arbitrary = frequency [(1, return Nothing)
                        , (3, liftM Just arbitrary)]
```

Generaatorid

Täisarvude genereerimine (ver. 1)

$$\begin{aligned}&\textbf{instance } \mathit{Arbitrary\ Int} \textbf{ where}\\ &\quad \mathit{arbitrary} = \mathit{choose}\ (-20, 20)\end{aligned}$$

Generaatorid

Täisarvude genereerimine (ver. 1)

$$\textbf{instance}\ \mathit{Arbitrary}\ \mathit{Int}\ \textbf{where}$$
$$\quad \mathit{arbitrary} = \mathit{choose}\ (-20, 20)$$

Täisarvude genereerimine (ver. 2)

$$\textbf{instance}\ \mathit{Arbitrary}\ \mathit{Int}\ \textbf{where}$$
$$\quad \mathit{arbitrary} = \mathit{sized}\ (\lambda n \rightarrow \mathit{choose}\ (-n, n))$$

Generaatorid

Rekursiivsete andmetüüpide genereerimine (ver. 1)

```
data Tree a = Leaf a
            | Node (Tree a) (Tree a)
instance Arbitrary a ⇒ Arbitrary (Tree a) where
  arbitrary = frequency [(1, liftM  Leaf  arbitrary)
                        ,(2, liftM2 Node arbitrary arbitrary)]
```

Generaatorid

Rekursiivsete andmetüüpide genereerimine (ver. 1)

```
data Tree a = Leaf a
            | Node (Tree a) (Tree a)
instance Arbitrary a ⇒ Arbitrary (Tree a) where
  arbitrary = frequency [(1, liftM  Leaf  arbitrary)
                        ,(2, liftM2 Node arbitrary arbitrary)]
```

Probleem

Termineeruvus pole garanteeritud!!

Generaatorid

Rekursiivsete andmetüüpide genereerimine (ver. 2)

```
instance Arbitrary a ⇒ Arbitrary (Tree a) where
  arbitrary = sized arbitraryTree

arbitraryTree    :: Arbitrary a ⇒ Int → Gen (Tree a)
arbitraryTree 0  = liftM  Leaf  arbitrary
arbitraryTree n = frequency [(1, liftM   Leaf  arbitrary)
                            ,(4, liftM2 Node t t)]
  where t = arbitraryTree (n `div` 2)
```

Generaatorid

Rekursiivsete andmetüüpide genereerimine (ver. 2)

```
instance Arbitrary a ⇒ Arbitrary (Tree a) where
  arbitrary = sized arbitraryTree

arbitraryTree    :: Arbitrary a ⇒ Int → Gen (Tree a)
arbitraryTree 0  = liftM  Leaf  arbitrary
arbitraryTree n  = frequency [(1, liftM   Leaf  arbitrary)
                             ,(4, liftM2  Node t t)]
  where t = arbitraryTree (n `div` 2)
```

NB!

- Teises võrrandis on ka võimalus puulehe genereerimiseks!
- Muidu genereeriksime ainult balanseeritud puid.

QuickCheck

Kokkuvõte

- Kuna Haskell on laisk keel, siis võimaldab ta kasutada ka lõpmatuid väärtusi; samas, omadused tohivad inspekteerida ainult lõplikku osa.
- *QuickCheck* suudab genereerida automaatselt ka funktsionaalseid väärtusi. Selleks peab tüüp kuuluma klassi *CoArbitrary*.
- Lisaks on *QuickCheck*-is (versioonis 2) ka vahendid efektidega arvutuste (sh. *IO*) testimiseks.