121. April 2015

Kartesisches Produkt Berechnen Sie mit Papier und Bleistift die Menge des jeweiligen kartesischen Produktes. Wenn Sie möchten, können Sie ihre Antworten mit GHCI überprüfen.

Hinweis: im Gegensatz zu Listen ist die Reihenfolge der Elemente einer Menge unerheblich (und Mengen enthalten auch keine "doppelten" Elemente).

1. $\{11, 7\} \times \{\clubsuit, \heartsuit\}$ $\big\{(11, \clubsuit), (11, \heartsuit), (7, \clubsuit), (7, \heartsuit)\big\}$
2. $(\{1, 2\} \times \{3, 4\}) \times \{5, 6\}$ $\big\{(1, 3, 5), (1, 3, 6), (1, 4, 5), (1, 4, 6), (2, 3, 5), (2, 3, 6), (2, 4, 5), (2, 4, 6)\big\}$
3. $\{1, 2\} \times (\{3, 4\} \times \{5, 6\})$ Abgesehen von der inneren Klammerung ist das Ergebnis identisch zur vorherigen Teilaufgabe. In der Mathematik ist es oft üblich, nicht zwischen diesen beiden Produkten zu unterscheiden und gleich Tripel zu verwenden, d.h. obwohl das kartesische Produkt eigentlich nicht assoziativ ist, wird es oft als assoziativ behandelt. Funktionale Sprachen unterscheiden jedoch üblicherweise zwischen solchen isomorphen Produkten, denn `(1,(2,3))` hat einen anderen Typ als `((1,2),3)` oder `(1,2,3)`.
4. $\{1, 2\} \times \{\}$ Das Produkt ist hier die leere Menge.
5. Sei A eine Menge mit n Elementen. Wie viele Elemente gibt es in $A \times \{27, 69\}$? Doppelt so viele, da jedes Element aus A zwei Mal im Produkt mit $\{27, 69\}$ vorkommt; also insgesamt $2n$.

AuswertungBerechnen Sie möglichst mit Papier und Bleistift den Wert, zu dem der gegebene Ausdruck jeweils auswertet. Überprüfen Sie Ihr Ergebnis mit GHCI.

1. `'c':'o':'o':'l':"!" "cool!"`
2. `(4 == 5.0):[] [False]`
3. `let mond="käse" in if mond=="käse" && 1==2 then False else 1+1==2 True`
4. ```
   [ z | c <- "grotesk", c/='k', c/='r', c/='e' && c/='s'
       , let z = if c=='o' then 'u' else c ]
   ```
   `"gut"`
5. `[(a,b) | a<-[1..5], b<-[5..1], a/=b] []` *Zur Erinnerung:* Wenn man herunterzählen möchte, dann muss man die Schrittweite angeben. Die Liste `[5..1]` ist leer, die Liste `[5,4..1]` enthält dagegen 5 Zahlen.
6. `(\x->"nope!") [c | c <- "yes!"] "nope!"` Der erste Teil definiert eine anonyme Funktion, welches Ihr Argument ignoriert und immer den String `"nope!"` zurückliefert. Der hintere Teil dieses Ausdrucks wertet wieder zu dem String `"yes!` aus.

Substitutionsmodell Werten Sie folgenden Ausdruck gemäß dem in der Vorlesung behandelten Substitutionsmodell aus:

$$\Big(\big(x((y : \_)x - y)\big)\ (3 + 4)\Big)\ (\mathit{tail}\ [1, 2])$$

Wir verzichten hier gleich auf alle überschüssige Klammern gemäß der Klammerkonvention.
   ```

Weiterhin rufen wir uns vorher noch die Definition von tail aus Foliensatz 2 (oder auch aus der Standardbibliothek) in Erinnerung: $(tail)\ (\_ : t) = t$. Damit man besser verstehen kann, wie das Pattern-Matching bei der Funktionsanwendung von *tail* und der anonymen Funktion abläuft, schreiben wir anstatt $[1, 2]$ gleich $(1 : (2 : [\,]))$, beide Schreibweisen sind ja äquivalent.

$$
\begin{aligned}
& (x((y : \_)x - y))\ \underline{(3 + 4)}\ (tail\ (1 : (2 : [\,]))) \\
\rightsquigarrow\ & (x((y : \_)x - y))\ 7\ \underline{(tail\ (1 : (2 : [\,])))} \\
\rightsquigarrow\ & (x((y : \_)x - y))\ 7\ (2 : [\,]) \\
\rightsquigarrow\ & ((y : \_)7 - y))\ (2 : [\,]) \\
\rightsquigarrow\ & 7 - 2 \rightsquigarrow 5
\end{aligned}
$$

Pattern-MatchingSchreiben Sie eine Funktion `myAnd`,[1] welche den Typ `Bool -> (Bool -> Bool)` hat und die logische Operation "und" implementiert. Zur Übung der Vorlesungsinhalte sollen Sie diese Aufgabe gleich drei Mal lösen, jeweils mit einer anderen Einschränkung:

1. Verwenden Sie bei der Definition ausschließlich Pattern-Matching.
2. Verwenden Sie bei der Definition ausschließlich Wächter (keine konstanten Patterns).
3. Verwenden Sie weder Pattern-Matching noch Wächter.

Natürlich dürfen Sie in keinem Fall die Operatoren aus der Standardbibliothek verwenden!

1. Da der Typ vorgegeben war, können wir diese Aufgabe rein mechanisch lösen, indem wir für jeden möglichen Input eine Zeile mit dem Ergebnis hinschreiben:

```
myAnd1 :: Bool -> Bool -> Bool
myAnd1 False False  = False
myAnd1 False True   = False
myAnd1 True  False  = False
myAnd1 True  True   = True
```

   Wer sich etwas Schreibarbeit sparen will, kann Fälle mit dem gleichen Ergebnis zusammenfassen. Für Haskell macht das keinen Unterschied, aber für uns ist es wohl leichter lesbar. Der einzige Spezialfall `True True` schreiben wir an erster Stelle, damit dieser zuerst geprüft wird. Da alle restlichen Fälle auf `False` abgebildet werden, können wir diese nun pauschal mit einem Wildcard Pattern abarbeiten, welche immer matched:

```
myAnd2 True True  = True
myAnd2 _          = False
```

   Eine alternative Lösung mit einem Pattern Match weniger wäre:

[1] Zur Vermeindung von Namenkonflikten mit der Standardbibliothek nicht "and" verwenden

```
myAnd3 True    y  = y
myAnd3 _          = False
```

2. ```
   myAnd4 x y
     | x, y      = True
     | otherwise = False
   ```

3. ```
   myAnd6 x y = if x then y else False
   ```

33
[keine]Typen0Welchen Typ haben folgenden Ausdrücke?

1. `['a','b','c']` `String` oder `[Char]` (Typsynonym)
2. `('a','b','c')` `(Char,Char,Char)`
3. `[(False,[1]),(True,[(2.0)])]` `[(Bool, [Double])]`
4. `([True,True],('z','o','o'))` `([Bool], (Char, Char, Char))`
5. `(\x -> ('a':x,False,([x])))` `[Char] -> ([Char], Bool, [[Char]])`
6. `[(\x y-> (x*2,y-1)) m n | m <- [1..5], even m, n <- [6..10]]` `[(Integer, Integer)]`

Hinweis: Kontrollieren Sie Ihre Antworten anschließend selbst mit GHCI! Auch wenn wir Typinferenz in der Vorlesung noch nicht behandelt haben, sollten Sie in der Lage sein, die meisten dieser einfachen Aufgaben bereits alleine mit Papier und Bleistift lösen zu können. Dies Aufgabe ist eigentlich auch einfacher als Aufgabe **??**!

33
[PDF]Substitutionsmodell II3Gegeben sind folgende zwei Funktionsdefinitionen

```
const x y =  x
negate x  = -x
```

und ein Ausdruck *const const* (*negate* 1) (*negate* 2) 3

1. In dem gegebenen Ausdruck müssen wir die implizite Klammerkonvention berücksichtigen, siehe Folie 01-46. Geben Sie den Ausdruck noch einmal mit vollständiger, expliziter Klammerung an!

   Funktionsanwendung ist implizit links geklammert, d.h.

$$\Big(\big((const\ const)\ (negate\ 1)\big)\ (negate\ 2)\Big)\ 3$$

[1] Test

2. Werten Sie nun den Ausdruck schrittweise gemäß dem in der Vorlesung behandelten Substitutionsmodell vollständig aus; unterstreichen Sie jeweils den bearbeiteten Teilausdruck.

   Eine mögliche Auswertung:

$$\underline{const\ const\ (negate\ 1)}\ (negate\ 2)\ 3 \rightsquigarrow \underline{const\ (negate\ 2)\ 3} \rightsquigarrow \underline{negate\ 2} \rightsquigarrow -2$$

   Wir dürfen aber auch eine andere Reihenfolge wählen, z.B.:

$$const\ const\ (\underline{negate\ 1})\ (negate\ 2)\ 3 \rightsquigarrow const\ const\ (-1)\ (\underline{negate\ 2})\ 3$$
$$\rightsquigarrow \underline{const\ const\ (-1)}\ (-2)\ 3 \rightsquigarrow \underline{const\ (-2)\ 3} \rightsquigarrow -2$$

1 Punkt auf die korrekte Klammerung; 2 Punkte auf eine vollständig korrekte Auswertung, dabei pro Fehler 1 Punkt abziehen (also hier: 1 Fehler = 1 Punkt), keine Trostpunkte.

Bei fehlerhafte Klammerung im ersten Teil können unter Beachtung des Folgesfehler trotzdem noch die vollen 2 Punkte auf dem zwetien Teil erzielt werden!

33

[hss]Modellieren mathematischer Funktion3Eine Ellipse um einen Mittelpunkt M ist gegeben durch ihre große Halbachse a und ihre kleine Halbachse b. Auf der großen Achse liegen die beiden Brennpunkte F_1, F_2 der Ellipse. Den Abstand e dieser Brennpunkte vom Mittelpunkt nennt man die Exzentrizität der Ellipse.

Die Exzentrizität hat einen Wert zwischen 0 und a und ist ein Maß dafür, wie „länglich“ die Ellipse ist. Die Exzentrizität hat den Wert 0 im Fall $a = b$, wenn also die Ellipse zu einem Kreis mit Radius a entartet. Die Exzentrizität hat den Wert a im Fall $b = 0$, wenn also die Ellipse zu einer Strecke der Länge $2a$ entartet.

Ellipse mit Halbachsen a und b mit $a \geq b \geq 0$	
Flächeninhalt:	πab
Exzentrizität:	$\sqrt{a^2 - b^2}$
Umfang:	$\pi\left(\frac{3}{2}(a+b) - \sqrt{ab}\right)$

In der obigen Tabelle sind die Formeln zur Berechnung von Flächeninhalt, Exzentrizität und Umfang angegeben.

Schreiben Sie eine Datei `H1-3.hs`, welche diese Funktionen implementiert:

```
flaecheninhalt :: Double -> Double -> Double
exzentrizitaet :: Double -> Double -> Double
umfang         :: Double -> Double -> Double
```

Hinweise

- Zur Vereinfachung müssen Sie nicht überprüfen, dass die Bedingung $a \geq b \geq 0$ tatsächlich eingehalten wird.

- Folgende Funktionen aus der Standardbibliothek dürfen Sie zusätzlich zu den üblichen mathematischen Operationen `+`,`-`,`*`,`/`,`^` verwenden: `pi`, `sqrt`. Schlagen Sie diese Funktionen ggf. in der Dokumentation der Standardbibliothek nach.
- *Beispiel* Eine Funktion zur Berechnung des Flächeninhalts eines Kreises, könnte man in Haskell wie folgt definieren:

```
kreisfläche :: Double -> Double
kreisfläche r = pi * r * r
```

- Punktabzug, falls Funktionsnamen, Signatur oder Dateiname nicht wie angegeben sind.

ellipse.hs 3 Punkte wenn alle automatischen Tests bestanden sind und alles korrekt ist. 2 Punkte bei 1–2 Fehlern oder auch schon falls nicht alle Tests bestanden werden (z.B. bei Mißachtung des Abgabeformats). 1 Trostpunkt für einen erkennbar bemühten Lösungsansatz.

Abgabe: Lösungen zu den Hausaufgaben können bis Dienstag, den 28.04.2015, 11:00 Uhr mit UniworX abgegeben werden.

Aufgrund des Klausurbonus müssen die Hausaufgaben von Ihnen alleine gelöst werden. Abschreiben bei den Hausaufgaben gilt als Betrug und kann zum Ausschluss von der Klausur zur Vorlesung führen.