

Einführung in die funktionale Programmierung

Wintersemester 2021/2022

Aufgabenblatt Nr. 4

Abgabe: Montag 29. November 2021, Abgabe bis 13.12.2021

Aufgabe 1 (35 Punkte)

Ziel dieser Aufgabe ist es eine einfache Version von „Conways Spiel des Lebens“^{1 2} in Haskell zu implementieren. **Ein Teil des Quellcodes ist bereits auf der Webseite der Veranstaltung zu finden.**

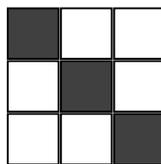
Das Spielfeld wird dargestellt durch *quadratische Matrizen*, deren Einträge entweder belegt (bevölkert), oder nicht belegt sind (ein Beispiel ist in Abbildung (a) zu sehen). Ein Eintrag an Position (i, j) in einer Matrix wird durch seine Zeile i und sein Spalte j bezeichnet, wobei die Zählung der Zeilen und Spalten bei 0 beginnt (wie für ein Beispiel in Abbildung (b) zu sehen ist). Solche Matrizen lassen sich in Haskell durch die folgende Datenstruktur darstellen:

```
data Matrix = Matrix [[Bool]] deriving(Eq)
```

Ist in der i -ten Liste der *Matrix* der j -te Eintrag `True`, so gilt der Eintrag (i, j) der Matrix als belegt. Beispielsweise entspricht

```
Matrix [[True,False,False],[False,True,False],[False,False,True]]
```

der Matrix aus Abbildung (a).



(a)

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)
(2,0)	(2,1)	(2,2)

(b)

Um solche Matrizen gut lesbar anzuzeigen, kann man folgende `Show`-Instanz verwenden:

```
instance Show Matrix where
  show (Matrix m) = unlines $ map showRow m
  where showRow = map (\x -> if x then '#' else ' ')
```

Machen Sie sich mit dem Programmcode auf der Webseite vertraut:

- Eine Haskell-Funktion des Typs: `Matrix -> (Int, Int) -> Bool`, die eine Matrix und eine Matrix-Position als Eingabe erwartet und die Belegung des durch die Koordinate gegebenen Matrix-Eintrags berechnet.
- Eine Haskell-Funktion `nachbarn :: (Int, Int) -> Int -> [(Int, Int)]` bestehend aus einer *List-Comprehension*, die eine Matrix-Position und die Dimension einer Matrix erwartet und die Positionen aller benachbarten Felder berechnet.
Beispielsweise liefert `nachbarn (1,1) 3` als Resultat die Liste `[(0,0), (0,1), (0,2), (1,0), (1,2), (2,0), (2,1), (2,2)]`.

¹http://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens

²John H. Conway ist vor kurzem, am 11. April 2020, an Komplikationen durch Covid-19 gestorben

- c) Eine Haskell-Funktion `anzahlBelegterNachbarn :: (Int, Int) -> Matrix -> Int`, die in einer gegebenen Matrix zu einem durch seine Position gegebenen Feld die Anzahl seiner belegten Nachbarn berechnet. Beispielsweise liefert der Aufruf von `anzahlBelegterNachbarn` mit der Koordinate `(1,1)` und der Matrix aus Abbildung (a), als Resultat 2.

Implementieren und testen Sie mit sinnvollen Beispielen:

- a) (10 Punkte) Implementieren Sie in Haskell eine Funktion `zugGOL :: Matrix -> Matrix`, die den Nachfolger einer Matrix gemäß der Spielregeln des Spiel des Lebens berechnet:
- Für eine Matrix-Zelle, die belegt ist:
 - * Jede Zelle mit einem oder keinem Nachbarn stirbt.
 - * Jede Zelle mit vier oder mehr Nachbarn stirbt.
 - * Jede Zelle mit zwei oder drei Nachbarn überlebt.
 - Für eine Matrix-Zelle, die leer ist:
 - * Jede Zelle mit drei Nachbarn wird belegt.

Hier bedeutet „stirbt“: nicht belegt in der Matrix (`False`); und „überlebt“: belegt in der Matrix (`True`).

Ruft man `zugGOL` auf dem Spielfeld von Abbildung (a) auf, soll das Resultat Matrix `[[False,False,False],[False,True,False],[False,False,False]]` sein.

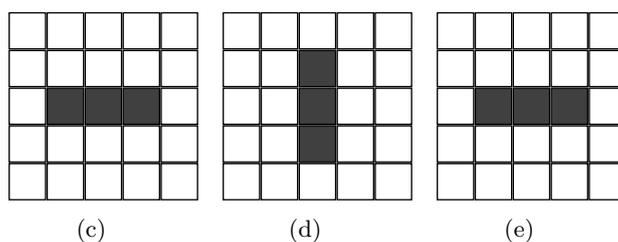
- b) (15 Punkte) Implementieren Sie eine Funktion `zuege :: Matrix -> Int -> [Matrix]`, die zu einer Anfangsmatrix `M` eine Liste der Folgematrizen berechnet, die durch `i` aufeinanderfolgende Spielzüge im Spiel des Lebens aus `M` entstehen.

Wenn `M` das Spielfeld aus Abbildung (a) ist, sollte `zuege M 2` als Resultat ergeben:

```
Matrix [[False,False,False],[False,True,False],[False,False,False]],
Matrix [[False,False,False],[False,False,False],[False,False,False]]
```

- c) (10 Punkte) Implementieren Sie eine Funktion `oszilliert :: Matrix -> Bool`, die zu einer gegebenen Matrix testet, ob der durch sie beschriebene Spielzustand oszilliert. Ein Spielzustand `M oszilliert`, wenn `M` nach `i`-Spielzügen ($i \geq 2$) wieder erreicht ist.

Im Beispiel unten ist ein oszillierender Zustand zu sehen: Beginnend bei Matrix (c) ist nach 2 Zügen im Spiel des Lebens wieder der Startzustand erreicht.



Aufgabe 2 (15 Punkte)

Die Haskell Funktion `lookup :: Eq a => a -> [(a, b)] -> Maybe b` erwartet einen Schlüssel `k` vom Typ `a` und eine Liste von Paaren vom Typ `(a,b)` und liefert `Just e` wenn `(k, e)` in der Liste enthalten ist, anderenfalls ist das Ergebnis `Nothing`.

Definieren sie in Haskell unter Verwendung der `foldr` Funktion eine Variante `myLookup`, die die Funktionalität von `lookup` besitzt. Ihre Funktion soll auch für unendliche Listen terminieren, d.h. `myLookup 4 [(x,y) | x <- [1..], y <- "A"]` soll als Ergebnisse `Just 'A'` liefern.

Begründen Sie, warum eine Implementierung von `myLookup`, die für unendliche Listen terminiert, mit Hilfe von `foldl` nicht möglich ist.

(15 Punkte)