Universität Bremen

Praktische Informatik 3 WS 14/15

10. Übungsblatt

Ausgabe: 06.01.15
Abgabe: 16.01.15

Christoph Lüth
Sandor Herms
Daniel Müller
Jan Radtke
Henrik Reichmann
Sören Schulze
Felix Thielke

10.1 *Haskell Gewinnt?* *20 Punkte*

In diesem Übungsblatt wollen wir eine Haskell-Version des bekannten Spieles *Vier gewinnt* (*Connect Four*) implementieren. Als Referenz für die Spielregeln gilt für uns der Wikipedia-Eintrag[1]:

> Das Spiel wird auf einem senkrecht stehenden hohlen Spielbrett gespielt, in das die Spieler abwechselnd ihre Spielsteine fallen lassen. Das Spielbrett besteht aus sieben Spalten (senkrecht) und sechs Reihen (waagerecht). Jeder Spieler besitzt 21 gleichfarbige Spielsteine. Wenn ein Spieler einen Spielstein in eine Spalte fallen lässt, besetzt dieser den untersten freien Platz der Spalte. Gewinner ist der Spieler, der es als erster schafft, vier oder mehr seiner Spielsteine waagerecht, senkrecht oder diagonal in eine Linie zu bringen. Das Spiel endet unentschieden, wenn das Spielbrett komplett gefüllt ist, ohne dass ein Spieler eine Viererlinie gebildet hat.

```
+-+-+-+-+-+-+-+          +-+-+-+-+-+-+-+
| | | | | | | |          | | | | | | | |
+-+-+-+-+-+-+-+          +-+-+-+-+-+-+-+
| | | | | | | |          | | | | |X| | |
+-+-+-+-+-+-+-+          +-+-+-+-+-+-+-+
| | | | |O| | |          | | | | |O| | |
+-+-+-+-+-+-+-+          +-+-+-+-+-+-+-+
| | |X| |O| |X|          | | |X| |O|O|X|
+-+-+-+-+-+-+-+          +-+-+-+-+-+-+-+
|O|X|X|O|X|X|O|          |O|X|X|O|X|X|O|
+-+-+-+-+-+-+-+          +-+-+-+-+-+-+-+
|X|O|O|X|O|O|X|          |X|O|O|X|O|O|X|
+-+-+-+-+-+-+-+          +-+-+-+-+-+-+-+
 1 2 3 4 5 6 7            1 2 3 4 5 6 7
Your move? 6
```

Abbildung 1: Ausgabe des Spielfeldes in überzeugendem Retro-Look.

Der Spielverlauf ist wie folgt:

1. Zu Beginn einer Runde wird der aktuelle Stand des Spieles in einer ansprechenden ASCII-Grafik ausgegeben, und der Benutzer um seinen Zug gebeten (Abb. 1 links). Als Zug wird die Nummer der betreffenden Spalte angegeben; dort fällt dann der Stein in die letzte freie Zeile hinein.

   Nach dem Zug des Benutzers rechnet Haskell seinen Zug aus, und eine neue Runde startet. Wenn beispielsweise der Benutzer als Zug 6 angibt, und Haskell Spalte 5 wählt, ergibt sich im nächsten Zug Abb. 1 rechts.

2. Vor jedem Zug muss geprüft werden, ob das Brett voll ist; in dem Fall wird das Spiel mit einem Unentschieden beendet.

3. Nach jedem Zug muss geprüft werden, ob der ziehende Spieler (Benutzer oder Haskell) gewonnen hat, d.h. vier Steine in einer Zeile, Spalte oder Diagonalen erzielt hat. Ist dies nicht der Fall, geht das Spiel in die nächste Runde.

[1] `http://de.wikipedia.org/wiki/Vier_gewinnt`

4. Die Benutzereingabe muss auf Plausibilität geprüft werden; ungültige Eingaben sollten in keinem Fall zum Programmabbruch führen. Die Ausgabe des Programmes sollte ähnlich wie in Abb. 1 aussehen.

Hinweise:

1. Es ist zweckmäßig, den Zustand des Spielbrettes als eine `Map (Int, Int) Player` darzustellen, wobei `Player` die beiden Spieler sind (`X` oder `O`).
2. Der Computerspieler (Haskell) sollte mindestens eine unmittelbar (im nächsten Zug bevorstehende) Niederlage verhindern, oder einen unmittelbar erreichbaren Siegeszug durchführen können. (In unserem Beispiel in Abb. 1 hätte also Haskell die Spalte 6 auswählen sollen, um die bevorstehende Niederlage zu verhindern.) Kann kein Zug berechnet werden, sollte zufällig ein möglicher Zug ausgewählt werden.

Punkteverteilung:

- Modellierung des Spielfeldes einschließlich Ausgabe *5 Punkte*
- Grundlegender Spielablauf *10 Punkte*
- Computerspieler *5 Punkte*

? *Verständnisfragen*

1. Warum ist die Erzeugung von Zufallszahlen eine Aktion?
2. Warum sind Aktionen nicht explizit als Zustandsübergang modelliert, sondern implizit als abstrakter Datentyp `IO`?
3. Was ist (außer dem Mangel an referentieller Transparenz) die entscheidende Eigenschaft, die Aktionen von reinen Funktionen unterscheidet?