Universität Bremen

Praktische Informatik 3 WS 14/15

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

4. Übungsblatt

Ausgabe: 11.11.14
Abgabe: 22.11.14

4.1 *Excel-lent** *20 Punkte*

Tabellenkalkulationsprogramme sind unbestritten ein großer Erfolg der modernen Informatik. Ohne sie wären komplizierte steuersparende Geschäftsbilanzen, sybillinische Controllingtabellen und waghalsige Finanzkonstruktionen, die halbe Weltwirtschaft an den Rand des Zusammebruches bringen können, gar nicht erst denkbar. Leider ist die Software des Marktführers nicht immer fehlerfrei, und schon gar nicht kostengünstig.

Deshalb wollen wir in dieser Aufgabe selber ein kleines Tabellenkalkulationsprogram entwickeln. Es verwaltet einen sogenannte *spread sheet*, d.h. eine zweidimensional indizierte Tabelle, die entweder beliebige Zeichenketten, Zahlen, oder Formeln enthält. Eine Formel kann dabei insbesondere auf andere Werte der Tabelle Bezug nehmen, und aus ihnen neue Werte berechnen.

Die Tabellenkalkulation wird in vier einfachen Schritten erstellt:

1. Eine Tabelle besteht aus Zellen. Eine Zelle enthält entweder

   - nichts (i.e. ist leer), oder
   - ein *label* (eine beliebige Zeichenkette), oder
   - einen formatierten Ausdruck. Hierbei wird ein Ausdruck wie folgt gebildet:
     - Addition, Subtraktion, Multiplikation oder Division zweier Ausdrücke;
     - Addition oder Multiplikation über Teile von Zeilen oder Spalten, also zum Beispiel Summe über die Spalten eins bis drei in Zeile 5, oder Multiplikation über die Zeilen null bis fünf in Spalte drei;
     - eine konstante Zahl (Double);
     - eine Referenz auf eine andere Zelle in Zeile i und Spalte j. Es ist legal, auf eine Zelle zu verweisen, die leer ist, oder einen nichtnumerischen Wert enthält; in dem Fall ist der Wert der Referenz null. Zyklische Verweise sind auch legal; in diesem Fall terminiert die Auswertung nicht.[1]
     - Zusätzlich kann ein Format angegeben werden, das für die Ausgabe eine Anzahl Nachkommastellen und ein Dezimalpunkt festlegt.

   Definieren Sie zwei algebraische Datentypen Cell und Expr, die den Inhalt der Zellen der Tabelle, und darin erlaubte Ausdrücke modellieren.

   Definieren Sie danach einen Datentyp Spreadsheet, der eine Tabelle modelliert, entweder direkt als Abbildung

   ```
   type Spreadsheet = ((Int, Int) → Cell, Int, Int)
   ```

   (wobei die beiden ganzen Zahlen die Größe der Tabelle darstellen), oder als Liste von Listen.

2. Definieren Sie Funktionen

   ```
   empty :: Spreadsheet
   set   :: (Int, Int) → Cell → Spreadsheet → Spreadsheet
   get   :: Spreadsheet → (Int, Int) → Int → String
   ```

   die eine leere Tabelle erstellen, eine bestimmte Zelle der Tabelle setzen, oder eine bestimmte Stelle auswerten.

[1] Das ist ein Feature, kein Bug: Der Benutzer wollte es eben so.

Der dritte Parameter in get ist die zur Verfügung stehende Breite der Ausgabe. Die resultierende Zeichenkette soll genau diese Länge haben, indem kürzere Strings links mit Leerzeichen aufgefüllt werden, und bei längeren Strings sollen bei Label die Zeichenkette von links abgeschnitten, und bei Ausdrücken die Zelle mit ### gefüllt werden.

3. Damit wird jetzt eine Funktion

   prnt :: Int→ Spreadsheet → String

   implementiert, die eine Tabelle auswertet, und eine gut lesbar formatierte (d.h. alle Zellen auf die gleiche Breite formatierte) textuelle Repräsentation mit der im zweiten Argument angegebenen Breite zurückgibt.

   Hier eine Beispielausgabe:

```
*Excel> putStrLn (prnt 45 testsheet)
---------------------------------------------
|              |       Übung 1|            16|
---------------------------------------------
|              |       Übung 2|            19|
---------------------------------------------
|              |       Übung 3|             5|
---------------------------------------------
|              |       Übung 4|            10|
---------------------------------------------
|              |    Durchschn.|         12.50|
---------------------------------------------
```

   Hier wird in Zelle (5,3) der Durchschnitt aus den Zellen 1 bis 3 der Spalte 3 berechnet.

4. Verkaufen — das Übungsblatt wird gerade rechtzeitig zum Weihnachtsgeschäft fertig.

Hinweise: Für die Auswertung ist es hilfreich, eine Funktion

```
eval :: Spreadsheet-> (Int, Int)-> Double
```

zu definieren, die den Wert der Tabelle an einer Stelle auswertet. Diese Funktion wird eine Hilfsfunktion haben, die rekursiv über den algebraischen Typen Expr definiert ist, und einen Ausdruck auswertet.

Um Listen von Koordinaten als Paare (i, j) zu erzeugen, die in einer Stelle (z.B. i) konstant sind, kann die Funktion zip zusammen mit der Aufzählung und der Funktion repeat verwendet werden:

```
zip :: [a] -> [b] -> [(a,b)]
repeat :: a -> [a]
Prelude> zip (repeat 3) [2..5]
[(3,2),(3,3),(3,4),(3,5)]
```

? *Verständnisfragen*

1. Was kennzeichnet einfach rekursive Funktionen, wie wir sie in der Vorlesung kennengelernt haben, und wie sind sie durch die Funktion foldr darstellbar?
2. Welche anderen geläufigen Funktionen höherer Ordnung kennen wir?
3. Was ist η-Kontraktion, und warum ist es zulässig?
4. Wann verwendet man foldr , wann foldl , und unter welchen Bedingungen ist das Ergebnis das gleiche?