Christoph Lüth
Bernd Gersdorf
Christian Maeder
René Wagner
Dennis Walter
Diedrich Wolter
Simon Ulbricht

10. Übungsblatt

Ausgabe: 27.01.11
Abgabe: 07.02.11

10.1 *Layout von Bäumen* *20 Punkte*

Die Informatik erscheint dem Laien oft als Dschungel. Überall Bäume! Im Besonderen trifft dies auf die funktionale Programmierung zu. Es ist bemerkenswert, dass diese sich gleichzeitig hervorragend eignet, um Bäume zu präsentieren, d.h. visuell aufzubereiten. Abb. 1 veranschaulicht dies beispielhaft.

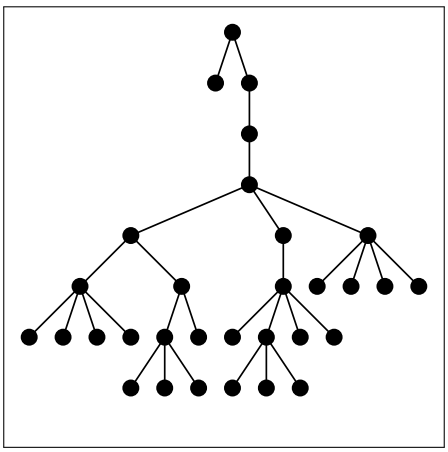

Abbildung 1: Ein funktional erzeugtes, Platz sparendes Baumlayout.

Implementieren Sie den von Kennedy [1, Abschn. 2-5] beschriebenen Algorithmus zur ansprechenden Darstellung von Bäumen in Haskell[1]. Erweitern Sie hierzu die Vorgaben aus der Vorlage. Verwenden Sie folgenden Datentypen für Bäume mit beliebig vielen Kindknoten:

```
data Tree a = Node a [Tree a]
```

Ein *Baumlayout* ist dann ein `Tree (a, Double)`, der für jeden Knoten dessen horizontale Verschiebung relativ zur Position seines Elternknotens enthält. Wir fordern fünf Eigenschaften von Baumlayouts:

- Alle Knoten einer Baumebene werden auf derselben horizontalen Linie (Höhe) dargestellt.
- Jeder Elternknoten soll zentriert über seinen Kindern stehen.
- Knoten derselben Ebene müssen zueinander einen Mindestabstand einhalten.
- Das Layout der Spiegelung eines Baumes `t` (als `mirrorTree t`) soll in der Darstellung spiegelsymmetrisch zum Layout von `t` sein. Hieraus folgt die wünschenswerte Eigenschaft, dass Bäume, die strukturell ihrem `mirrorTree` gleichen, achsensymmetrisch dargestellt werden. Die QuickCheck-Property `prop_sym` formalisiert diese Forderung.
- Strukturell gleichartige Teilbäume sollen unabhängig von ihrer Lage im Gesamtbaum gleich dargestellt werden (s. Abb. 2 in [1])

Der Algorithmus arbeitet «bottom-up» und berechnet neben dem Layout auch die *horizontale Ausdehnung* eines Baums als Paare $(x_{\min}, x_{\max})$ für all seine Ebenen:

```
type Extent = [(Double, Double)]
```

[1] Die Referenzimplementierung im Papier verwendet die strikte funktionale Sprache *Standard ML*.

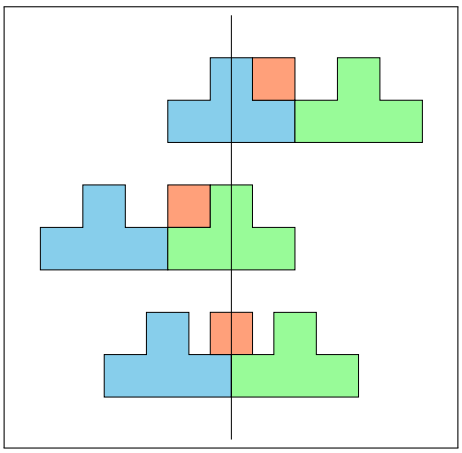

Abbildung 2: Zentrieren von Ausdehnungen durch links-orientierte (oben) und rechts-orientierte (Mitte) Anordnung mit anschließender Mittelwertbildung (unten).

Die Arbeit erfolgt in drei Schritten:

i) Gegeben einen Baum `Node l ts`, berechne Layout und Ausdehnungen für die Kindbäume `ts`.

ii) Arrangiere die Kindbäume anhand ihrer Ausdehnungen möglichst nah nebeneinander, ohne ihre Struktur zu verändern. Die fertigen Kind-Layouts werden also lediglich horizontal verschoben.

iii) Platziere den Elternknoten (`l`) zentriert über seine Kindknoten.

Wir bezeichnen nachfolgend mit x die Position eines Kindbaums relativ zu seinem Elternknoten. Die Hauptschwierigkeit besteht in der Berücksichtigung der Symmetrieeigenschaft. Die Lösung liegt darin, die Kindbäume zunächst links-orientiert aneinander zu reihen, d.h. mit dem ersten (linken) Kindbaum bei $x = 0$ beginnend die folgenden Bäume rechts ($x > 0$) anzuhängen; dann rechts-orientiert das letzte (rechte) Kind auf $x = 0$ zu positionieren und die Geschwister links ($x < 0$) anzufügen; und schließlich den Mittelwert der sich so ergebenden Korrekturen zu verwenden. Abb. 2 veranschaulicht dies.

Die Implementierung kann über die vorgegebenen QuickCheck-Properties getestet werden. Ihre Aufgaben in der Zusammenfassung:

1. Vervollständigen Sie die Funktionen `merge`, `mergeList`, `fitListL/R` und `design`. (12 Punkte)
2. Definieren Sie einige Bäume `testTree<i>`, deren Layout beispielhaft nachweist, dass die oben angegebenen fünf Eigenschaften vom Algorithmus respektiert werden. (5 Punkte)
3. Speichern Sie diese Layouts zur visuellen Kontrolle mithilfe der bereitgestellten Funktion `writeSVG` als skalierbare Vektorgrafik (SVG).[2] [3] (3 Punkte)

Gutes Gelingen!

Literatur

[1] Andrew Kennedy. Drawing trees. *Journal of Functional Programming*, 6:527–534, 1996. doi: 10.1017/S0956796800001830. URL `http://research.microsoft.com/~akenn/fun/DrawingTrees.pdf`.

[2] Sie benötigen hierzu die auf der PI3-Webseite verfügbare Kombinatorbücherei `TinySVG.hs` zur Erzeugung von SVG-Grafiken.

[3] Moderne Browser können `.svg` Dateien anzeigen, ebenso wie das freie Zeichenprogramm *Inkscape*, das auch einen PDF-Export bietet. Eine Alternative ist das *Rasterizer*-Modul des *Batik* Toolkits (unter `http://xmlgraphics.apache.org/batik/`).