Teil VII

Algorithmik I

14. Listen versus Arrays

- Zwei verschiedene Containertypen: Listen und Arrays
- Unterschiedliche Stärken und Schwächen:
  - Listen: flexibel — Erweiterung in konstanter Zeit (*Cons* aka '::');
  - Arrays: wahlfreier Zugriff in konstanter Zeit (*nth* aka $a.[i]$).
- Best of both worlds: flexible Arrays
  - Erweiterung in logarithmischer Zeit (*Cons* aka '::');
  - wahlfreier Zugriff in logarithmischer Zeit (*nth* aka $a.[i]$).
- Erste Implementierungsidee: Binärbäume

14. Wahlfreier Zugriff

- Welcher Knoten im Binärbaum korrespondiert zu welchem Index?

$$\begin{tikzcd}
\text{Listen} \arrow[r, shift right, "\textit{from-list}"'] & \text{Binärbäume} \arrow[l, shift right, "\textit{to-list}"']
\end{tikzcd}$$

- *Idee:* die Zuordnung wird indirekt festgelegt, indem wir einen Baum auf eine Liste abbilden.
- *Spezifikation* des wahlfreier Zugriffs:

$$\mathit{tree}.[i] = (\mathit{to\text{-}list}\ \mathit{tree}).[i] \tag{12a}$$

$$(\mathit{from\text{-}list}\ \mathit{list}).[i] = \mathit{list}.[i] \tag{12b}$$

  Wenn $\mathit{to\text{-}list}\ (\mathit{from\text{-}list}\ x) = x$, dann folgt (12b) aus (12a). (Beachte: wir überladen die Notation $\mathit{xs}.[i]$!)
- *Gesucht:* eine geschickte Definition von *to-list*!

14. Exploring the design space

Erste unstrittige (?) Festlegung: das Wurzelelement ist das erste Element, das mit der Hausnummer 0. Aus diesem Grund setzen wir das Wurzelelement vor den linken Teilbaum.

```
type Tree ⟨'elem⟩
   | Leaf
   | Node of 'elem ∗ Tree ⟨'elem⟩ ∗ Tree ⟨'elem⟩
```

Implementierung von *to-list*:

```
let rec to-list = function
   | Leaf          → []
   | Node (a, l, r) → a :: combine (to-list l, to-list r)
```

☞ Im Wesentlichen müssen wir festlegen, wie die beiden Listen für die Teilbäume kombiniert werden. Lassen wir uns damit etwas Zeit …

14. Herleitung von *nth*

Referenzimplementierung: *nth* auf Listen.

$$
\begin{array}{l}
\textbf{\textit{let rec}}\ nth\ (list, i) = \\
\quad \textbf{\textit{match}}\ list\ \textbf{\textit{with}} \\
\quad\ \mid [\,] \qquad\ \ \to None \\
\quad\ \mid x :: xs \to \textbf{\textit{if}}\ i = 0\ \textbf{\textit{then}}\ Some\ x\ \textbf{\textit{else}}\ nth\ (xs, i-1)
\end{array}
$$

Herleitung von *nth* auf Binärbäumen. **Fall** $tree = Leaf$**:**

$$
\begin{array}{cl}
 & nth\ (Leaf, i) \\
= & \quad \{\ \text{Spezifikation (12a)}\ \} \\
 & nth\ (to\text{-}list\ Leaf, i) \\
= & \quad \{\ \text{Definition von}\ to\text{-}list\ \} \\
 & nth\ ([\,], i) \\
= & \quad \{\ \text{Definition von}\ nth\ \} \\
 & None
\end{array}
$$

14. Herleitung von *nth*

Fall $\mathit{tree} = \mathit{Node}\ (a, l, r)$ **und** $i = 0$**:**

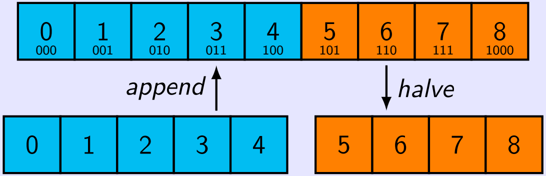
$$
\begin{aligned}
& \mathit{nth}\ (\mathit{Node}\ (a, l, r), 0) \\
= \quad & \{\ \text{Spezifikation (13)}\ \} \\
& \mathit{nth}\ (\mathit{to\text{-}list}\ (\mathit{Node}\ (a, l, r)), 0) \\
= \quad & \{\ \text{Definition von}\ \mathit{to\text{-}list}\ \} \\
& \mathit{nth}\ (a :: \mathit{combine}\ (\mathit{to\text{-}list}\ l, \mathit{to\text{-}list}\ r), 0) \\
= \quad & \{\ \text{Definition von}\ \mathit{nth}\ \} \\
& \mathit{Some}\ a
\end{aligned}
$$

14. Herleitung von *nth*

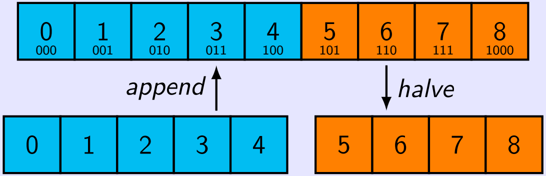
Fall $\mathit{tree} = \mathit{Node}\ (a, l, r)$ **und** $i > 0$**:**

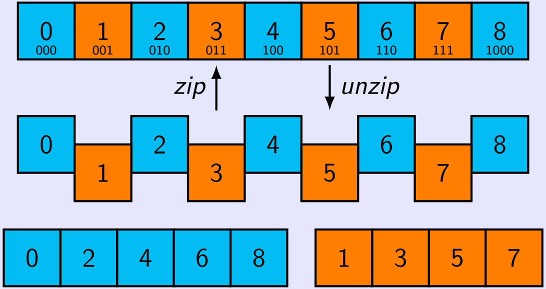
$$
\begin{aligned}
& \mathit{nth}\ (\mathit{Node}\ (a, l, r), i) \\
= \quad & \{\ \text{Spezifikation (13)}\ \} \\
& \mathit{nth}\ (\mathit{to\text{-}list}\ (\mathit{Node}\ (a, l, r)), i) \\
= \quad & \{\ \text{Definition von } \mathit{to\text{-}list}\ \} \\
& \mathit{nth}\ (a :: \mathit{combine}\ (\mathit{to\text{-}list}\ l, \mathit{to\text{-}list}\ r), i) \\
= \quad & \{\ \text{Definition von } \mathit{nth} \text{ und } i > 0\ \} \\
& \mathit{nth}\ (\mathit{combine}\ (\mathit{to\text{-}list}\ l, \mathit{to\text{-}list}\ r), i - 1) \\
= \quad & \{\ ?\ \}
\end{aligned}
$$

☞ Wir müssen festlegen, wie die beiden Listen für die Teilbäume kombiniert werden. Was ist eine geschickte Wahl?


0
000
1
001
2
010
3
011
4
100
5
101
6
110
7
111
8
1000
append
halve
0
1
2
3
4
5
6
7
8

0
000
1
001
2
010
3
011
4
100
5
101
6
110
7
111
8
1000
zip
unzip
0
1
2
3
4
5
6
7
8
0
2
4
6
8
1
3
5
7


0 000
1 001
2 010
3 011
4 100
5 101
6 110
7 111
8 1000
append
halve
0
1
2
3
4
5
6
7
8


0 000
1 001
2 010
3 011
4 100
5 101
6 110
7 111
8 1000
zip
unzip
0
1
2
3
4
5
6
7
8
0
2
4
6
8
1
3
5
7

14. Herleitung von *nth*

Eigenschaften von *nth*:

$$\begin{aligned}
\mathit{nth}\,(\mathit{append}\,(\mathit{xs}, \mathit{ys}), i) &= \textbf{\textit{if}}\ \ i < \mathit{length}\ \mathit{xs}\ \textbf{\textit{then}}\ \mathit{nth}\,(\mathit{xs}, i) \\
&\phantom{= \textbf{\textit{if}}\ \ i < \mathit{length}\ \mathit{xs}\ }\textbf{\textit{else}}\ \ \mathit{nth}\,(\mathit{ys}, i - \mathit{length}\ \mathit{xs}) \\
\mathit{nth}\,(\mathit{zip}\,(\mathit{xs}, \mathit{ys}), i) &= \textbf{\textit{if}}\ \ i \,\%\, 2 = 0\ \textbf{\textit{then}}\ \mathit{nth}\,(\mathit{xs}, i \div 2) \\
&\phantom{= \textbf{\textit{if}}\ \ i \,\%\, 2 = 0\ }\textbf{\textit{else}}\ \ \mathit{nth}\,(\mathit{ys}, i \div 2)
\end{aligned}$$

☞ Verwenden wir *append*, benötigen wir Informationen über die Länge der Teillisten bzw. Größe der Teilbäume.

☞ Die Gleichung für *zip* setzt voraus, dass $0 \leqslant \mathit{length}\ \mathit{xs} - \mathit{length}\ \mathit{ys} \leqslant 1$. (Warum?)
Aber: Kenntnis der konkreten Längen ist nicht nötig.

14. Herleitung von *nth*

Fall $\mathit{tree} = \mathit{Node}\ (a, l, r)$ **und** $i > 0$:

$$
\begin{aligned}
& \mathit{nth}\ (\mathit{zip}\ (\mathit{to\text{-}list}\ l, \mathit{to\text{-}list}\ r), i - 1) \\
= \quad & \{\ \text{Eigenschaft, siehe oben}\ \} \\
& \mathbf{if}\ (i - 1)\ \%\ 2 = 0\ \mathbf{then}\ \mathit{nth}\ (\mathit{to\text{-}list}\ l, (i - 1) \div 2)\ \mathbf{else} \ldots \\
= \quad & \{\ \text{Spezifikation (13)}\ \} \\
& \mathbf{if}\ (i - 1)\ \%\ 2 = 0\ \mathbf{then}\ \mathit{nth}\ (l, (i - 1) \div 2)\ \mathbf{else} \ldots
\end{aligned}
$$

14. Implementierung von *nth*

Wir erhalten:

$$
\begin{array}{l}
\textbf{let rec } nth\ (tree, i) = \\
\quad \textbf{match } tree \textbf{ with} \\
\quad\ \mid Leaf \qquad\qquad \to None \\
\quad\ \mid Node\ (x, l, r) \to \textbf{if } \quad i = 0 \qquad\qquad\quad \textbf{then } Some\ x \\
\qquad\qquad\qquad\qquad\quad \textbf{elif } (i-1) \% 2 = 0\ \textbf{then } nth\ (l, (i-1) \div 2) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textbf{else } \ nth\ (r, (i-1) \div 2)
\end{array}
$$

bzw.

$$
\begin{array}{l}
\textbf{let rec } nth\ (tree, i) = \\
\quad \textbf{match } tree \textbf{ with} \\
\quad\ \mid Leaf \qquad\qquad \to None \\
\quad\ \mid Node\ (x, l, r) \to \textbf{if } \quad i = 0 \qquad\quad \textbf{then } Some\ x \\
\qquad\qquad\qquad\qquad\quad \textbf{elif } i \% 2 = 1\ \textbf{then } nth\ (l, i \div 2) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textbf{else } \ nth\ (r, i \div 2 - 1)
\end{array}
$$

14. Implementierung von *nth*

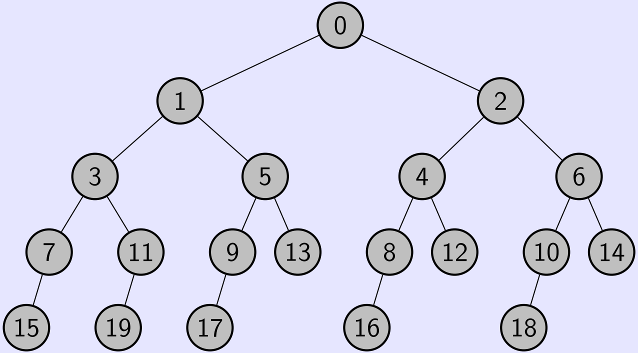
Wir erhalten:

$$
\begin{array}{l}
\textbf{let rec } nth\ (tree, i) = \\
\quad \textbf{match } tree \textbf{ with} \\
\quad \mid Leaf \qquad\qquad \to None \\
\quad \mid Node\ (x, l, r) \to \textbf{if } \ i = 0 \qquad\qquad\qquad \textbf{then } Some\ x \\
\qquad\qquad\qquad\qquad\quad \textbf{elif } (i-1)\ \%\ 2 = 0 \ \textbf{then } nth\ (l, (i-1) \div 2) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textbf{else } \ nth\ (r, (i-1) \div 2)
\end{array}
$$

bzw.

$$
\begin{array}{l}
\textbf{let rec } nth\ (tree, i) = \\
\quad \textbf{match } tree \textbf{ with} \\
\quad \mid Leaf \qquad\qquad \to None \\
\quad \mid Node\ (x, l, r) \to \textbf{if } \ i = 0 \qquad\quad \textbf{then } Some\ x \\
\qquad\qquad\qquad\qquad\quad \textbf{elif } i\ \%\ 2 = 1 \ \textbf{then } nth\ (l, i \div 2) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \textbf{else } \ nth\ (r, i \div 2 - 1)
\end{array}
$$

14. Nummerierung



14. Spezifikation von *cons*

Spezifikation der Erweiterung:

$$\mathit{to\text{-}list}\ (\mathit{cons}\ (a, \mathit{tree})) = a :: \mathit{to\text{-}list}\ \mathit{tree} \tag{13}$$

Ziel: einen Ausdruck e mit $\mathit{to\text{-}list}\ (\mathit{cons}\ (a, \mathit{tree})) = \mathit{to\text{-}list}\ e$ herleiten, so dass e als Definition von $\mathit{cons}\ (a, \mathit{tree})$ dienen kann.

14. Herleitung von *cons*

Fall $tree = Leaf$:

$$
\begin{aligned}
& \mathit{to\text{-}list}\ (\mathit{cons}\ (a, \mathit{Leaf}))\\
= \quad & \quad \{\ \text{Spezifikation (13)}\ \}\\
& a :: \mathit{to\text{-}list}\ \mathit{Leaf}\\
= \quad & \quad \{\ \text{Definition von } \mathit{zip}\ \}\\
& a :: \mathit{zip}\ ([\,], \mathit{to\text{-}list}\ \mathit{Leaf})\\
= \quad & \quad \{\ \text{Definition von } \mathit{to\text{-}list}\ \}\\
& a :: \mathit{zip}\ (\mathit{to\text{-}list}\ \mathit{Leaf}, \mathit{to\text{-}list}\ \mathit{Leaf})\\
= \quad & \quad \{\ \text{Definition von } \mathit{to\text{-}list}\ \}\\
& \mathit{to\text{-}list}\ (\mathit{Node}\ (a, \mathit{Leaf}, \mathit{Leaf}))
\end{aligned}
$$

14. Herleitung von *cons*

Fall $tree = Node\ (b, l, r)$:

$$
\begin{aligned}
& \mathit{to\text{-}list}\ (\mathit{cons}\ (a, Node\ (b, l, r))) \\
= \quad & \{\ \text{Spezifikation (13)}\ \} \\
& a :: \mathit{to\text{-}list}\ (Node\ (b, l, r)) \\
= \quad & \{\ \text{Definition von}\ \mathit{to\text{-}list}\ \} \\
& a :: b :: \mathit{zip}\ (\mathit{to\text{-}list}\ l, \mathit{to\text{-}list}\ r) \\
= \quad & \{\ \text{Definition von}\ \mathit{zip}\ \} \\
& a :: \mathit{zip}\ (b :: \mathit{to\text{-}list}\ r, \mathit{to\text{-}list}\ l) \\
= \quad & \{\ \text{Spezifikation (13)}\ \} \\
& a :: \mathit{zip}\ (\mathit{to\text{-}list}\ (\mathit{cons}\ (b, r)), \mathit{to\text{-}list}\ l)) \\
= \quad & \{\ \text{Definition von}\ \mathit{to\text{-}list}\ \} \\
& \mathit{to\text{-}list}\ (Node\ (a, \mathit{cons}\ (b, r), l))
\end{aligned}
$$

14. Implementierung von *cons*

Wir erhalten:

$$
\begin{array}{l}
\textbf{\textit{let rec}}\ cons\ (a, tree) = \\
\quad \textbf{\textit{match}}\ tree\ \textbf{\textit{with}} \\
\quad \mid Leaf \qquad\qquad \to Node\ (a, Leaf, Leaf) \\
\quad \mid Node\ (b, l, r) \to Node\ (a, cons\ (b, r), l)
\end{array}
$$

14. Invariante

Invariante: für jeden Knoten *Node* (a, l, r) gilt, dass der linke Teilbaum l genau so viele Elemente wie der rechte Teilbaum r enthält oder ein Element mehr.

$$0 \leqslant \mathit{size}\ l - \mathit{size}\ r \leqslant 1$$

☞ Anderenfalls arbeitet *nth* nicht korrekt.

Die Erweiterung *cons* (a, tree) erhält die Invariante.

$$\mid \mathit{Node}\ (b, l, r) \rightarrow \mathit{Node}\ (a, \mathit{cons}\ (b, r), l)$$

Enthalten l und r gleich viele Elemente, so ist der linke Teilbaum nach dem Einfügen um eins größer. Umgekehrt: war l vor dem Einfügen um eins größer, dann sind nach dem Einfügen beide Teilbäume gleich groß.

14. Braun Bäume

Die resultierenden Bäume heißen *Braun Bäume*.

Damit haben wir übrigens Knobelaufgabe #13 gelöst.

Die Struktur eines Braun Baums ist durch die Anzahl der Elemente eindeutig festgelegt. Ist zum Beispiel die Gesamtzahl gleich $2^k - 1$, dann ist der Braun Baum vollständig ausgeglichen.