Teil VII

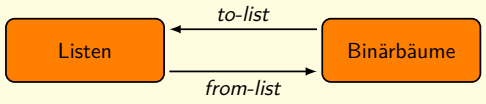
Algorithmik I

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?



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

$$tree.[i] = (\textit{to-list}\ tree).[i] \tag{12a}$$
$$(\textit{from-list}\ list).[i] = list.[i] \tag{12b}$$

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

```
let rec nth (list, i) =
  match list with
    | []      → None
    | x :: xs → if i = 0 then Some x else nth (xs, i − 1)
```

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

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

14. Herleitung von *nth*

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

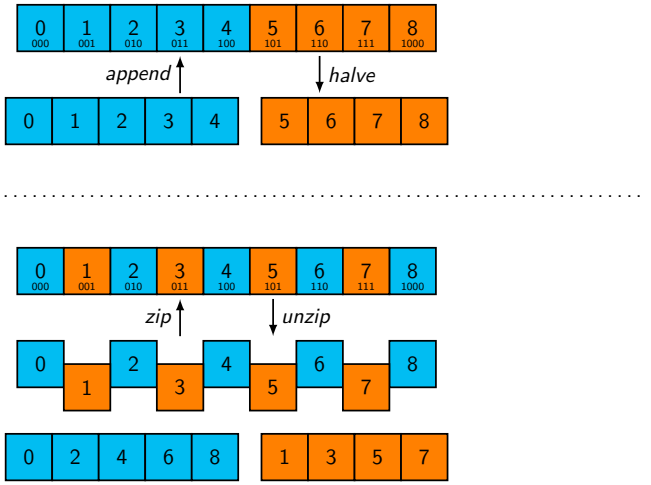
$$\begin{aligned} & nth\ (Node\ (a, l, r), 0) \\ = & \quad \{ \text{ Spezifikation (13) } \} \\ & nth\ (\textit{to-list}\ (Node\ (a, l, r)), 0) \\ = & \quad \{ \text{ Definition von } \textit{to-list} \ \} \\ & nth\ (a :: combine\ (\textit{to-list}\ l, \textit{to-list}\ r), 0) \\ = & \quad \{ \text{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

14. Herleitung von *nth*

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

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

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



14. Herleitung von *nth*

Eigenschaften von *nth*:

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

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

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

14. Herleitung von *nth*

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

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

14. Implementierung von *nth*

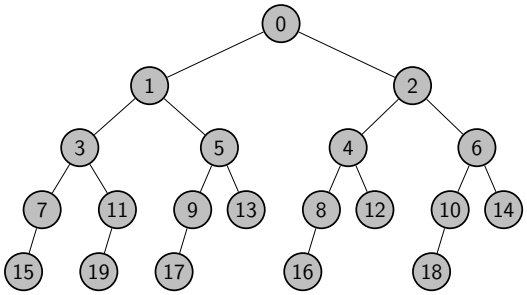
Wir erhalten:

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

bzw.

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

14. Nummerierung



14. Spezifikation von *cons*

Spezifikation der Erweiterung:

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

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 $\mathit{tree} = \mathit{Leaf}$:

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

14. Herleitung von *cons*

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

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

14. Implementierung von *cons*

Wir erhalten:

```
let rec cons (a, tree) =
  match tree with
    | Leaf           → Node (a, Leaf, Leaf)
    | Node (b, l, r) → Node (a, cons (b, r), l)
```

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.

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

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.