

# Das “Countdown”-Problem

RALF HINZE

Institut für Informatik III

Universität Bonn

Email: [ralf@informatik.uni-bonn.de](mailto:ralf@informatik.uni-bonn.de)

Homepage: <http://www.informatik.uni-bonn.de/~ralf/>

Juni 2002

# Überblick

---

- ✘ Lösungsansatz
- ✘ Darstellung von Ausdrücken
- ✘ “top down”-Generierung von Kandidaten
- ✘ “bottom up”-Generierung von Kandidaten
- ✘ Kombinatorische Funktionen
- ✘ Bestimmung des besten Kandidaten
- ✘ Verbesserungen

# “Generate and select” . . .

*Grundidee:* Generiere Kandidaten; wähle unter den Kandidaten den besten aus.

$$\begin{aligned} \textit{solve} & \quad :: \quad [Expr] \rightarrow Int \rightarrow Expr \\ \textit{solve es n} & = \textit{closest n (generate es)} \end{aligned}$$

## . . . “Generate and select”

*Verallgemeinerung:* statt Zahlen geben wir Teilausdrücke vor.

```
type Choices a = [a]
generate      :: [Expr] → Choices Expr
closest      :: Int → Choices Expr → Expr
```

*Vereinfachung:* wir nehmen zunächst an, daß alle übergebenen Ausdrücke *genau* einmal verwendet werden müssen.

# “top down”-Generierung von Kandidaten . . .

Strukturell rekursiv (Kochrezept):

$$\begin{aligned} \textit{generate} [e] &= [e] \\ \textit{generate} (e : es) &= \dots (\textit{generate} es) \dots \end{aligned}$$

*Problem:* wie läßt sich eine Teillösung zu einer Gesamtlösung erweitern?

# . . . “top down”-Generierung von Kandidaten

“divide and conquer”:

$$\begin{aligned} \textit{generate } [e] &= [e] \\ \textit{generate } es &= [t \mid (s_1, s_2) \leftarrow \textit{split2 } es, \\ &\quad l \leftarrow \textit{generate } s_1, \\ &\quad r \leftarrow \textit{generate } s_2, \\ &\quad t \leftarrow \textit{combine } l \ r] \end{aligned}$$

# “bottom-up” Generierung von Kandidaten

$$\begin{aligned} \text{generate } [e] &= [e] \\ \text{generate } es &= \text{concat } [\text{generate } es' \mid es' \leftarrow \text{step } es] \end{aligned}$$

Die Funktion *step* wählt zwei Ausdrücke aus und kombiniert diese zu einem neuen Ausdruck.

$$\text{step} :: [Expr] \rightarrow \text{Choices } [Expr]$$

Für  $es' \leftarrow \text{step } es$  gilt  $\text{length } es' + 1 = \text{length } es$ .

# Überblick

---

- ✓ Lösungsansatz
- ✗ Darstellung von Ausdrücken
- ✗ “top down”-Generierung von Kandidaten
- ✗ “bottom up”-Generierung von Kandidaten
- ✗ Kombinatorische Funktionen
- ✗ Bestimmung des besten Kandidaten
- ✗ Verbesserungen

# Darstellung von Ausdrücken . . .

```
data Expr = Con{ val :: Int }  
          | Add{ val :: Int, left :: Expr, right :: Expr }  
          | Sub{ val :: Int, left :: Expr, right :: Expr }  
          | Mul{ val :: Int, left :: Expr, right :: Expr }  
          | Div{ val :: Int, left :: Expr, right :: Expr }
```

# . . . Darstellung von Ausdrücken . . .

Klevere Konstruktoren.

$(\langle + \rangle), (\langle - \rangle), (\langle * \rangle), (\langle / \rangle)$	$::$	$Expr \rightarrow Expr \rightarrow Expr$
$l \langle + \rangle r$	$=$	$Add (val\ l + val\ r) l\ r$
$l \langle - \rangle r$	$=$	$Sub (val\ l - val\ r) l\ r$
$l \langle * \rangle r$	$=$	$Mul (val\ l * val\ r) l\ r$
$l \langle / \rangle r$	$=$	$Div (val\ l\ 'div'\ val\ r) l\ r$

# . . . Darstellung von Ausdrücken

**instance** *Eq Expr* **where**

$t == u \quad = \quad \text{val } t == \text{val } u$

**instance** *Ord Expr* **where**

$t \leq u \quad = \quad \text{val } t \leq \text{val } u$

$\text{compare } t \ u \quad = \quad \text{compare } (\text{val } t) \ (\text{val } u)$

# Überblick

---

- ✓ Lösungsansatz
- ✓ Darstellung von Ausdrücken
- ✗ “top down”-Generierung von Kandidaten
- ✗ “bottom up”-Generierung von Kandidaten
- ✗ Kombinatorische Funktionen
- ✗ Bestimmung des besten Kandidaten
- ✗ Verbesserungen

# “top down”-Generierung von Kandidaten . . .

*Zunächst:* verwende jeden übergebenen Ausdruck *genau* einmal.

$$\begin{aligned} \text{gen} &:: [\text{Expr}] \rightarrow \text{Choices Expr} \\ \text{gen } [] &= [] \\ \text{gen } [e] &= [e] \\ \text{gen } s &= [t \\ & \quad | (s_1, s_2) \leftarrow \text{split2 } s \\ & \quad , l \leftarrow \text{gen } s_1 \\ & \quad , r \leftarrow \text{gen } s_2 \\ & \quad , t \leftarrow \text{combine } l \ r] \end{aligned}$$

# . . . “top down”-Generierung von Kandidaten . . .

Dann: kombiniere zwei Ausdrücke.

$$\begin{aligned} \textit{combine}, \textit{combine}' &:: \textit{Expr} \rightarrow \textit{Expr} \rightarrow \textit{Choices Expr} \\ \textit{combine } e_1 e_2 & \\ \quad | e_1 \leq e_2 &= \textit{combine}' e_1 e_2 \\ \quad | \textit{otherwise} &= \textit{combine}' e_2 e_1 \end{aligned}$$

Die Hilfsfunktion  $\textit{combine}'$  geht davon aus, daß  $e_1 \leq e_2$ .

$$\begin{aligned} \textit{combine}' e_1 e_2 &= [e_1 \langle + \rangle e_2] \\ &\# [e_2 \langle - \rangle e_1 \mid \textit{val } e_2 > \textit{val } e_1] \\ &\# [e_2 \langle * \rangle e_1 \mid \textit{val } e_1 \neq 1] \\ &\# [e_2 \langle / \rangle e_1 \mid \textit{val } e_1 \neq 1, \textit{val } e_2 \textit{'mod'} \textit{val } e_1 == 0] \end{aligned}$$

## . . . “top down”-Generierung von Kandidaten

*Schließlich:* verwende jeden übergebenen Ausdruck *höchstens* einmal.

$$\begin{aligned} \textit{generate} &:: [\textit{Expr}] \rightarrow \textit{Choices Expr} \\ \textit{generate } s &= [t \mid s \leftarrow \textit{subsets1 } s, t \leftarrow \textit{gen } s] \end{aligned}$$

# Überblick

---

- ✓ Lösungsansatz
- ✓ Darstellung von Ausdrücken
- ✓ “top down”-Generierung von Kandidaten
- ✗ “bottom up”-Generierung von Kandidaten
- ✗ Kombinatorische Funktionen
- ✗ Bestimmung des besten Kandidaten
- ✗ Verbesserungen

# “bottom up”-Generierung von Kandidaten . . .

Nur die Implementierung von *gen* ist unterschiedlich.

$$\begin{aligned} \textit{gen} &:: [\textit{Expr}] \rightarrow \textit{Choices Expr} \\ \textit{gen} [] &= [] \\ \textit{gen} [e] &= [e] \\ \textit{gen} es &= \textit{concat} [\textit{gen} es' \mid es' \leftarrow \textit{step} es] \end{aligned}$$

# . . . “bottom up”-Generierung von Kandidaten

$$\begin{aligned} \textit{step} & \quad :: \quad [Expr] \rightarrow Choices [Expr] \\ \textit{step es} & = \quad [e : es_2 \\ & \quad | \quad (e_1, es_1) \leftarrow \textit{remove es} \\ & \quad , \quad (e_2, es_2) \leftarrow \textit{remove es}_1 \\ & \quad , \quad e_1 \leq e_2 \\ & \quad , \quad e \leftarrow \textit{combine}' e_1 e_2] \end{aligned}$$

# Überblick

---

- ✓ Lösungsansatz
- ✓ Darstellung von Ausdrücken
- ✓ “top down”-Generierung von Kandidaten
- ✓ “bottom up”-Generierung von Kandidaten
- ✗ Kombinatorische Funktionen
- ✗ Bestimmung des besten Kandidaten
- ✗ Verbesserungen

# Kombinatorische Funktionen . . .

Nicht-leere Teilmengen einer Menge (dargestellt als Listen).

```
subsets1           :: [a] → Choices [a]
subsets1 [a]       = [[a]]
subsets1 (a : as)  = s ++ [a] : map (a:) s
  where s           = subsets1 as
```

**Aufgabe:** Wie läßt sich *subsets1* verbessern (*Tip:* Reihenfolge der Teilmengen)?



## . . . Kombinatorische Funktionen

Entfernen eines Elements aus einer Liste.

$$\begin{aligned} \mathit{remove} & \quad :: \quad [a] \rightarrow \mathit{Choices} (a, [a]) \\ \mathit{remove} [] & \quad = \quad [] \\ \mathit{remove} (a : x) & \quad = \quad (a, x) : [(b, a : y) \mid (b, y) \leftarrow \mathit{remove} x] \end{aligned}$$

# Überblick

---

- ✓ Lösungsansatz
- ✓ Darstellung von Ausdrücken
- ✓ “top down”-Generierung von Kandidaten
- ✓ “bottom up”-Generierung von Kandidaten
- ✓ Kombinatorische Funktionen
- ✗ Bestimmung des besten Kandidaten
- ✗ Verbesserungen

# Bestimmung der besten Lösung

Verallgemeinerung: Verwendung einer Distanzfunktion.

$$\begin{aligned} \textit{closest} &:: (\textit{Ord } a) \Rightarrow (a \rightarrow \textit{Int}) \rightarrow \textit{Choices } a \rightarrow a \\ \textit{closest } f \textit{ as} &= \textit{snd } (\textit{minimum } [(f \textit{ a}, a) \mid a \leftarrow \textit{as}]) \end{aligned}$$

**Aufgabe:** Wie läßt sich *closest* verbessern?

Distanzfunktion für das “Countdown”-Problem.

$$\begin{aligned} \textit{dist} &:: \textit{Int} \rightarrow \textit{Expr} \rightarrow \textit{Int} \\ \textit{dist } n \textit{ t} &= \textit{abs } (n - \textit{val } t) \end{aligned}$$

# Überblick

---

- ✓ Lösungsansatz
- ✓ Darstellung von Ausdrücken
- ✓ “top down”-Generierung von Kandidaten
- ✓ “bottom up”-Generierung von Kandidaten
- ✓ Kombinatorische Funktionen
- ✓ Bestimmung des besten Kandidaten
- ✗ Verbesserungen

# Mögliche Verbesserungen

---

- ✘ Assoziativität ausnutzen: entweder  $a + (b + c)$  oder  $(a + b) + c$  als Kandidat generieren, aber nicht beide.
- ✘ “top down-Ansatz: Vermeide wiederholte Berechnungen (Memoisierung).
- ✘ “bottom up“-Ansatz: Suchraum gezielter durchsuchen.