Die Korrektheit von Mergesort

Christoph Lüth

11. November 2002

Definition von Mergesort

Die Funktion Mergesort ist wie folgt definiert:

```
msort :: [Int]-> [Int]
msort xs
  | length xs <= 1 = xs
  | otherwise = merge (msort front) (msort back) where
    (front, back) = splitAt ((length xs) `div` 2) xs
    merge :: [Int]-> [Int]-> [Int]
    merge [] x = x
    merge y [] = y
    merge (x:xs) (y:ys)
      | x<= y     = x:(merge xs (y:ys))
      | otherwise = y:(merge (x:xs) ys)
```

Zu zeigen ist die *Korrektheit von Mergesort*: die Rückgabe von Mergesort soll *sortiert* sein, und eine *Permutation* der Eingabeliste (d.h. dieselben Elemente in einer möglicherweise anderen Reihenfolge enthalten).

Diese beiden Eigenschaften definieren wir mit Hilfe folgender Haskell-Funktionen:

```
sorted :: [Int]-> Bool
sorted []  = True
sorted [x] = True
sorted (x:xs) = x <= head xs && sorted xs

count :: Int-> [Int]-> Int
count z []     = 0
count z (x:xs) =
  if z== x then 1+ count z xs
  else count z xs
```

Damit definieren wir

$$\texttt{perm}\, X\, Y \Leftrightarrow \forall z. \texttt{count}\, z\, X = \texttt{count}\, z\, Y$$

Wie wir sehen ist `perm` keine ausführbare Haskell-Funktion, weil wir über *alle möglichen Werte* von z (d.h. über alle ganzen Zahlen) quantifiziert haben. Das macht aber nichts, wir wollen `perm` ja schließlich nicht ausführen, sondern nur etwas damit beweisen.

Die Korrektheit von Mergesort (oder jeder anderen Sortierfunktion) ist damit folgende Gleichung:

$$\texttt{perm}\, X\, (\texttt{msort}\, X) \wedge \texttt{sorted}\, (\texttt{msort}\, X) \tag{1}$$

Eigenschaften von Permutation und Sortiertheit

Ersteinmal zeigen wir einen nützlichen Hilfssatz (Lemma) über die Funktion `count`:

Lemma 1

$$\texttt{count}\, z\, X +\!\!+\, Y = \texttt{count}\, z\, X + \texttt{count}\, z\, Y$$

Beweis: Der Beweis erfolgt durch Induktion über das erste Argument X, weil $+\!\!+$ durch primitive Rekursion über dieses Argument definiert ist.
Induktionsbasis:

$$\begin{array}{rcl} \texttt{count}\, z\, ([\,] +\!\!+ Y) & = & \texttt{count}\, z\, Y \\ & = & 0 + \texttt{count}\, z\, Y \\ & = & \texttt{count}\, z\, [\,] + \texttt{count}\, z\, Y \end{array}$$

Induktionssschritt: Die Induktionsvoraussetzung ist $\texttt{count} z (X +\!\!+ Y) = \texttt{count} z X + \texttt{count}\, z\, Y$. Wir unterscheiden zwei Fälle:

1.

$$\begin{array}{rcll} \texttt{count}\, z\, (x : X +\!\!+ Y) & = & 1 + \texttt{count}\, z\, (X +\!\!+ Y) & \text{Annahme: } x = z \\ & = & 1 + \texttt{count}\, z\, X + \texttt{count}\, z\, Y & \text{nach I.v.} \\ & = & \texttt{count}\, z\, (x : X) + \texttt{count}\, z\, Y & \text{Annahme: } x = z \end{array}$$

2.

$$\begin{array}{rcll} \texttt{count}\, z\, x : X +\!\!+ Y & = & \texttt{count}\, z\, X +\!\!+ Y & \text{Annahme: } x \neq z \\ & = & \texttt{count}\, z\, X + \texttt{count}\, z\, Y & \text{nach I.v.} \\ & = & \texttt{count}\, z\, x : X + \texttt{count}\, z\, Y & \text{Annahme: } x \neq z \end{array}$$

□

Im folgenden benutzen wir folgende *Notation*: Wir schreiben $X \sim Y$ für $\texttt{perm} X Y$. Das ist insbesondere nützlich, weil `perm` eine transitive und reflexive Relation ist, wie das folgende Lemma zeigt:

Lemma 2 *(i) Wenn $X \sim Y$ und $Y \sim Z$, dann $X \sim Z$*

(ii) $X \sim X$

Beweis (i):

$$\begin{array}{rl} & X \sim Y, Y \sim Z \\ \Leftrightarrow & \texttt{count}\, x\, X = \texttt{count}\, x\, Y, \texttt{count}\, x\, Y = \texttt{count}\, x\, Z \\ \Leftrightarrow & \texttt{count}\, x\, X = \texttt{count}\, x\, Z \\ \Leftrightarrow & X \sim Y \end{array}$$

Beweis (ii):

$$X \sim X \Leftrightarrow \texttt{count}\, x\, X = \texttt{count}\, x\, X$$

$\square$

Lemma 2 erlaubt uns Beweise über $\sim$ durch Verketten von Umformungen zu formulieren: um beispielsweise $X \sim Y$ zu zeigen, zeigen wir

$$\begin{array}{rcl} X & \sim & X_1 \\ & = & X_2 \\ & \sim & X_3 \\ & = & Y \end{array}$$

Natürlich ist $\sim$ auch symmetrisch, i.e. wenn $X \sim Y$, dann $Y \sim X$, aber das brauchen wir hier nicht. Wichtig ist aber, dass $\sim$ eine Kongruenz bezüglich der Listenkonkatenation $++$ ist:

Lemma 3 *Wenn $A \sim X$ und $B \sim Y$, dann $A ++ B \sim X ++ Y$*

Beweis:

$$\begin{array}{rcl} A \sim X, B \sim Y & \Leftrightarrow & \texttt{count}\, z_1\, A = \texttt{count}\, z_1\, X, \texttt{count}\, z_2\, B = \texttt{count}\, z_2\, Y \\ & \Rightarrow & \texttt{count}\, z\, A + \texttt{count}\, z\, B = \texttt{count}\, z\, X + \texttt{count}\, z\, Y \\ & \Leftrightarrow & \texttt{count}\, z\, (A ++ B) = \texttt{count}\, z\, (X ++ Y) \\ & \Leftrightarrow & A ++ B \sim X ++ Y \end{array}$$

$\square$

Nützliche Sonderfälle von Lemma 3 sind $A = X$, i.e. $Y \sim B$, dann $A ++ B \sim A ++ Y$, und analog $B = Y$. Wir benötigen ferner folgende Variation von Lemma 3:

Lemma 4

$$X ++ Y \sim Y ++ X$$

Der Beweis erfolgt analog zu Lemma 3.

Wir beschließend diesen Abschnitt mit einem Lemma über die Funktion `splitAt`:

Lemma 5 *Sei $\texttt{splitAt}\, n\, X = (Y, Z)$, dann ist $X = Y ++ Z$.*

Dieses ist weniger eine Eigenschaft als eher ein Teil der *Spezifikation* von `splitAt`. Der Beweis erfolgt durch Induktion über n, und bleibt dem Leser als Übung überlassen.

Die Funktion `merge`

Genauso wie bei Mergesort die eigentliche Arbeit von der Funktion `merge` erledigt wird, steckt die eigentliche Beweisarbeit für die Korrektheit von Mergesort in zwei Lemmata über die `merge` Funktion. `merge` ist durch Rekursion über zwei Parameter gleichzeitig definiert, also benutzen wir als analoges Beweisprinzip folgende Spezialisierung der Fixpunktinduktion.

Um eine Eigenschaft P über `merge` zu zeigen, d.h. $P(\texttt{merge} X Y)$ für ein beliebige X, Y, zeigen wir folgendes:

(i) Induktionsbasis: $P([], Y)$ und $P(X, [])$ für beliebige X, Y;

(ii) Induktionsschritt:
Wenn $P(x : X, Y)$ und $P(X, y : Y)$, dann $P(x : X, y : Y)$.

Lemma 6

$$\texttt{merge}\, a\, b \sim a +\!\!+\, b$$

Induktionsbasis: $\texttt{merge}\, a\, [\,] = a = a +\!\!+ [\,]$, und $\texttt{merge}\, [\,]\, b = b = [\,] +\!\!+\, b$.
Induktionsschritt: Wir betrachten $\texttt{merge}\, (a : as)\, (b : bs)$ und unterscheiden zwei Fälle:

1. $a \leq b$, dann ist

$$\begin{array}{rcll} \texttt{merge}\, (a : as)\, (b : bs) & = & a : (\texttt{merge}\, as\, (b : bs)) & \\ & \sim & a : (as +\!\!+ (b : bs)) & \text{nach I.v. und Lemma 3} \\ & \sim & a : as +\!\!+\, b : bs & \text{nach Def. } +\!\!+ \end{array}$$

2. $a > b$, dann ist

$$\begin{array}{rcll} \texttt{merge}\, (a : as)\, (b : bs) & = & b : (\texttt{merge}\, (a : as)\, bs) & \\ & \sim & b : (a : as +\!\!+\, bs)) & \text{nach I.v. und emma 3} \\ & \sim & a : as +\!\!+ [b] +\!\!+\, bs & \text{nach Lemma 4} \\ & \sim & a : as +\!\!+\, b : bs & \end{array}$$

□

Lemma 7 *Wenn* $\texttt{sorted}\, X, \texttt{sorted}\, Y$, *dann* $\texttt{sorted}\, (\texttt{merge}\, X\, Y)$.

Induktionsbasis:

- $\texttt{merge}\, X\, [\,] = X$, damit $\texttt{sorted}\, (\texttt{merge}\, X\, [\,])$ wenn $\texttt{sorted}\, X$.
- $\texttt{merge}\, [\,]\, Y = Y$, damit $\texttt{sorted}\, (\texttt{merge}\, [\,]\, Y)$ wenn $\texttt{sorted}\, Y$.

Induktionsschritt: Auch hier unterscheiden wir zwei Fälle:

1. $x \leq y$

$$\begin{array}{ll} & \texttt{sorted}\, (\texttt{merge}\, (x : X)\, (y : Y)) \\ \Leftrightarrow & \texttt{sorted}\, (x : \texttt{merge}\, X\, (y : Y)) \\ \Leftrightarrow & x \leq \texttt{head}\, (\texttt{merge}\, X\, (y : Y)) \wedge \texttt{sorted}\, (\texttt{merge}\, X\, (y : Y)) \quad \text{nach Def. } \texttt{sorted} \end{array}$$

Der zweite Teil der Konjunktion ist die Induktionsvoraussetzung. Für den ersten Teil haben wir

$$\texttt{head}\, (\texttt{merge}\, X\, Y) = \texttt{head}\, X \vee \texttt{head}\, (\texttt{merge}\, X\, Y) = \texttt{head}\, Y \qquad (2)$$

(Für $X = Y = [\,]$ sind beide Gleichungen undefiniert.) Damit ist $\texttt{head}(\texttt{merge}\, X\, (y : Y)) = y$, und nach Voraussetzung $x \leq y$; oder $\texttt{head}\, (\texttt{merge}\, X\, (y : Y)) = \texttt{head}\, X$, und dann ist $x < \texttt{head}\, X$, was wegen der Voraussetzung $\texttt{sorted}\, (x : X)$ gilt.

2. $x > y$, ist völlig analog:

$$\begin{array}{ll} & \texttt{sorted}\,\texttt{merge}\,(x:X)\,(y:Y) \\ \Leftrightarrow & \texttt{sorted}\,(y:\texttt{merge}\,(x:X)\,Y) \\ \Leftrightarrow & y \leq \texttt{head}\,(\texttt{merge}\,(x:X)\,Y) \wedge \texttt{sorted}\,(\texttt{merge}\,(x:X)\,Y) \quad \text{nach Def.} \end{array}$$

Der zweite Teil der Konjunktion ist die Induktionsvoraussetzung. Für den ersten Teil ist mit (2) entweder $\texttt{head}\,(\texttt{merge}\,(x:X)\,Y) = x$, und nach Voraussetzung $y < x$; oder $\texttt{head}\,(\texttt{merge}\,(x:X)\,Y) = \texttt{head}\,Y$, und dann ist $y < \texttt{head}\,Y$, was wegen der Voraussetzung $\texttt{sorted}\,(y:Y)$ gilt.

$\square$

Die Korrektheit von Mergesort

Wir können jetzt die Korrektheit von Mergesort zeigen, i.e. Gleichung (1) in zwei Teilen:

(i)
$$\texttt{perm}\,(\texttt{msort}\,X)\,X,$$

(ii)
$$\texttt{sorted}\,(\texttt{msort}\,X).$$

Das Beweisprinzip hierbei muß der Definition von Mergesort entsprechen. Mergesort ist rekursiv über der *Länge* der Liste definiert; die Basis sind Listen der Länge eins oder null, und die im Rekurssionsschritt wird die Länge der Liste halbiert. Deshalb zeigen wir die Behauptungen (i) und (ii) durch *Induktion über der Länge der Liste*[1]. Aus Gründen der Übersichtlichkeite zeigen wir beide Teile getrennt.

Beweis (i):

Induktionsbasis: Wenn $\texttt{length}\,X \leq 1$, dann gilt durch Einsetzen der Funktionsdefinition $\texttt{msort}\,X = X \sim X$.

Induktionsschritt: Zu zeigen:

$$\texttt{merge}\,(\texttt{msort}\,\mathit{front})\,(\texttt{msort}\,\mathit{back}) \sim X$$

mit $(\mathit{front}, \mathit{back}) = \texttt{splitAt}\,(\texttt{length}\,X/2)\,X$.

Die Induktionsannahme ist

$$\mathit{front} \sim \texttt{msort}\,\mathit{front},\ \mathit{back} \sim \texttt{msort}\,\mathit{back}$$

Nach Lemma 5 ist $X = \mathit{front} +\!+ \mathit{back}$. Damit ist

$$\begin{array}{rcll} X & \sim & \mathit{front} +\!+ \mathit{back} & \\ & \sim & \texttt{msort}\,\mathit{front} +\!+ \texttt{msort}\,\mathit{back} & \text{nach I.v.} \\ & \sim & \texttt{merge}\,(\texttt{msort}\,\mathit{front})\,(\texttt{msort}\,\mathit{back}) & \text{nach Lemma 6} \end{array}$$

Beweis (ii):

[1] Dieses ist keine *natürliche Induktion*, sondern tatsächlich eine Fixpunktinduktion.

Induktionsbasis: Wenn $\texttt{length}\,X \leq 1$, dann gilt durch Einsetzen der Funktionsdefinition $\texttt{msort}\ X = X$. Sowohl für $X = []$ als auch für $X = [x]$ folgt direkt $\texttt{sorted}\,X$, und damit $\texttt{sorted}\,(\texttt{msort}\,X)$

Induktionsschritt: Zu zeigen ist

$$\texttt{sorted}\,(\texttt{merge}\,(\texttt{msort}\,\mathit{front})\,(\texttt{msort}\,\mathit{back})).$$

Die Induktionssvoraussetzung ist

$$\texttt{sorted}\,(\texttt{msort}\,\mathit{front}), \texttt{sorted}\,(\texttt{msort}\,\mathit{front})$$

. Damit wird der Induktionsschritt direkt durch Lemma 7 bewiesen.

$\square$