Ludwig-Maximilians-Universität München
Department "Institut für Informatik"
Prof. Dr. Christian Böhm
Elke Achtert, Arthur Zimek

München, 03.11.2006

Informatik 1
WS 2006/07

Übungsblatt 3: Rekursive Funktionen

Besprechung: 13.11.–17.11.2006

Abgabe aller mit **Hausaufgabe** markierten Aufgaben bis Freitag, 10.11.2006, 18:00 Uhr

Aufgabe 3-1 *Wurzelziehen* **(Hausaufgabe)**

Keine Angst, es geht nicht um Zahnärzte, sondern nur um Mathematik.

Ein Näherungswert für die Quadratwurzel einer positiven reellen Zahl x kann nach einem Iterationsverfahren berechnet werden. Das Verfahren arbeitet mit Schätzwerten für $\sqrt{x}$, die in der Iteration schrittweise verbessert werden. Der erste Schätzwert für $\sqrt{x}$ ist stets $w = 1.0$, die Quadratwurzel aus x ist also der *Iterationswert* von 1.0.

Der *Iterationswert* von einem Schätzwert w für $\sqrt{x}$ wird bestimmt, indem man zunächst den *Fehler* $|w^2 - x|$ berechnet. Wenn der Fehler kleiner ist als ein vorgegebener Schwellwert ε, ist w ein hinreichend genauer Näherungswert für $\sqrt{x}$, das heißt, der Iterationswert von w ist w selbst. Andernfalls berechnet man die *Verbesserung* $\frac{1}{2}(w + \frac{x}{w})$ des Schätzwerts (die Verbesserung hat stets einen kleineren Fehler als w), und der Iterationswert von w ist der Iterationswert von dieser Verbesserung.

Für $\varepsilon = 0.001$ und $x = 2.0$ wird der Iterationswert also so berechnet:

w	*Fehler* $\lvert w^2 - x\rvert$	$< \varepsilon$?	*Verbesserung* $\frac{1}{2}(w + \frac{x}{w})$
1.0	1.0	nein	1.5
1.5	0.25	nein	1.41666666667
1.41666666667	0.00694444445389	nein	1.41421568627
1.41421568627	0.00000600729212685	ja, also Iterationswert 1.41421568627	

Auf der Webseite der Vorlesung finden Sie unter „Dateien" zu diesem Übungsblatt die Datei `3-1.sml` mit einer unvollständigen Implementierung dieses Verfahrens. Kopieren Sie sich die Datei in Ihr Unterverzeichnis für dieses Übungsblatt. Ergänzen Sie in Ihrer Kopie der Datei die Definitionen von `fehler(x,w)` und `verbesserung(x,w)` und `iterationswert(x,w)` und geben Sie die vervollständigte Datei `3-1.sml` als Lösung ab. Sie können voraussetzen, dass der Fall $x \leq 0$ nicht vorkommt.

Hinweis: Die obige Beschreibung zur Berechnung des Iterationswerts lässt sich unmittelbar in eine rekursive Definition übersetzen. Alle anderen Funktionen sind nicht-rekursiv. Sie können die Ergebnisse Ihrer Funktion mit denen von `Math.sqrt` vergleichen.

Die Definition der Funktion `wurzel` in der Datei `3-1.sml` gibt Ihnen eine wertvolle Anregung zur Lösung der Aufgaben 3-2 und 3-3.

Aufgabe 3-2 *Quadratzahltest* **(Hausaufgabe)**

Definieren Sie eine SML-Funktion `istQuadratzahl : int -> bool`, die für eine natürliche Zahl n bestimmt, ob n eine Quadratzahl ist.

Hinweis: Eine Funktion $\mathbb{N} \times \mathbb{N} \rightarrow$ **bool** , die zu einem Tupel $(n, k) \in \mathbb{N} \times \mathbb{N}$ entscheidet, ob n das Quadrat einer Zahl $i \in \mathbb{N}, i \leq k$ ist, könnte sich als hilfreich erweisen. Eine solche Hilfsfunktion definieren Sie am besten rekursiv.

Geben Sie Ihre Lösung in einer Datei `3-2.sml` ab.

Aufgabe 3-3 *Vollkommenheitstest* **(Hausaufgabe)**

Für eine Zahl $n \in \mathbb{N}$ heißt die Zahl $i \in \mathbb{N}$ ein *positiver echter Teiler* von n, wenn es eine Zahl $k \in \mathbb{N}, k > 1$ gibt, so dass $n = i \cdot k$ ist.

Eine natürliche Zahl $n \in \mathbb{N} \setminus \{0\}$ heißt *vollkommen*, wenn Sie gleich der Summe ihrer positiven echten Teiler ist. Beispiele:

- Die Zahl 6 ist vollkommen, da ihre positiven echten Teiler 1, 2 und 3 sind und $1 + 2 + 3 = 6$ gilt.
- Die Zahl 8 ist nicht vollkommen, da ihre positiven echten Teiler 1, 2 und 4 sind und $1 + 2 + 4 = 7 \neq 8$ gilt.

Definieren Sie eine SML-Funktion `istVollkommen : int -> bool`, die für eine natürliche Zahl $n \in \mathbb{N} \setminus \{0\}$ bestimmt, ob n eine vollkommene Zahl ist.

Hinweis: Für $i, n > 0$ gilt: i ist ein Teiler von n genau dann, wenn $n \bmod i = 0$. Die kleinsten vollkommenen Zahlen sind übrigens 6, 28, 496, 8128, 33550336. Wie man sieht, sind vollkommene Zahlen etwas sehr seltenes. Zahlen sind eben auch nur Menschen.

Geben Sie Ihre Lösung in einer Datei `3-3.sml` ab.

Aufgabe 3-4 *Nullstellenberechnung* **(Hausaufgabe)**

Eine näherungsweise Nullstelle einer stetigen Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ in einem Intervall $[a, b] \subseteq \mathbb{R}$ mit $a \leq b$ und $f(a) \cdot f(b) \leq 0$ kann mit der Intervallschachtelungsmethode bis auf einen Fehler $\varepsilon > 0$ folgendermaßen berechnet werden:

- Falls $b - a < \varepsilon$, so ist das Ergebnis a.
- Andernfalls bestimme die Intervallmitte $c = \frac{(a+b)}{2}$.
- Wenn $f(a) \cdot f(c) \leq 0$, suche eine Nullstelle von f im Intervall $[a, c]$.
- Andernfalls suche eine Nullstelle im Intervall $[c, b]$.

(a) Wir nehmen an, dass eine stetige Funktion `f : real -> real` bereits definiert ist. Definieren Sie eine SML-Funktion `nullstelleVonF : real * real * real -> real`, die für ein Tupel $(a, b, \varepsilon) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^+$ mit $a \leq b$ und $f(a) \cdot f(b) \leq 0$ einen Wert $x \in [a, b]$ bestimmt, der um höchstens ε von einer Nullstelle von `f` abweicht.

(b) Verallgemeinern Sie nun die Funktion in einer neuen SML-Funktion `nullstelle : (real -> real) * real * real * real -> real` (nicht erschrecken!), die als erstes Argument eine (stetige) Funktion erhält, für die die Nullstelle wie oben bestimmt wird. Sie bildet also ein Tupel $(f, a, b, \varepsilon) \in (\mathbb{R} \rightarrow \mathbb{R}) \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}^+$ auf eine Nullstelle von f zwischen a und b ab (mit einer Abweichung von höchstens ε).

Geben Sie die beiden Funktionen in einer Datei `3-4.sml` ab.

Aufgabe 3-5 *Rekursive Definition des Produkts* **(Hausaufgabe)**

Für diese Aufgabe stehen Addition, Subtraktion, Multiplikation und Division auf natürlichen Zahlen nicht zur Verfügung. Statt dessen sind (auf einer maschinennäheren Implementierungsebene) einfachere arithmetische Funktionen für natürliche Zahlen realisiert, die Sie benutzen können.

(a) `vorgaenger: int -> int`
Liefert Argument $-$ 1 für positives Argument. Nicht definiert, wenn das Argument ≤ 0 ist.

`summe: int * int -> int`
Liefert die Summe der Argumente. Nicht definiert, wenn ein Argument negativ ist.

Die Datei `3-5a.sml` definiert diese Funktionen. Kopieren Sie sich die Datei in Ihr Unterverzeichnis für dieses Übungsblatt.

Definieren Sie mit Hilfe dieser Funktionen eine Funktion `produkt(a,b)`, die das Produkt von zwei natürlichen Zahlen liefert. Negative Zahlen brauchen nicht behandelt zu werden. Die Funktion soll ganz analog zu der Definition von `potenz(a,b)` aufgebaut sein, die in der Vorlesung besprochen wurde. (Hinweis: für $b > 0$ gilt $a \cdot b = a + a \cdot (b-1)$)

Geben Sie die um Ihre Definition von `produkt` ergänzte Datei `3-5a.sml` ab.

(b) `ist_gerade: int -> bool`
true für geradzahliges Argument, sonst false. Nicht definiert, wenn das Argument negativ ist.

`haelfte: int -> int`
Liefert Argument dividiert durch 2. Nicht definiert, wenn Argument ungerade oder negativ ist.

`doppelt: int -> int`
Liefert Argument mal 2. Nicht definiert, wenn das Argument negativ ist.

Die Datei `3-5b.sml` definiert diese Funktionen zusätzlich zu denen der vorigen Teilaufgabe. Kopieren Sie sich die Datei in Ihr Unterverzeichnis für dieses Übungsblatt.

Definieren Sie mit Hilfe dieser Funktionen eine Funktion `produkt'(a,b)`, die das Produkt von zwei natürlichen Zahlen effizienter berechnet. Die Funktion soll ganz analog zu der Definition von `potenz'(a,b)` aufgebaut sein, die in der Vorlesung besprochen wurde.

Geben Sie die um Ihre Definition von `produkt'` ergänzte Datei `3-5b.sml` ab.

Aufgabe 3-6 *Zeitmessung der rekursiven Produkt-Funktionen*

Für diese Aufgabe geben Sie nichts ab, aber Sie können es am Rechner ausprobieren.

In der Vorlesung wurde der Zeitbedarf für die Funktionen `potenz(a,b)` und `potenz'(a,b)` mit mathematischen Mitteln geschätzt.

Für die Funktionen `produkt(a,b)` und `produkt'(a,b)` der vorigen Aufgabe wollen wir nun wirkliche Zeiten messen. In der Datei `3-6.sml` finden Sie einige Hilfsmittel für die Messung des Zeitbedarfs bei der Auswertung von Ausdrücken.

Sie können in einer Kopie dieser Datei Ihre Definitionen von `produkt(a,b)` und `produkt'(a,b)` am Ende eintragen und dann in SML die Aufrufe ausprobieren, die in dem Kommentar am Anfang der Datei stehen. (Der Rest der Datei geht über den derzeitigen Vorlesungsstoff hinaus.)

Welche Erkenntnisse ziehen Sie aus den Messwerten?

Achtung: Wenn Sie zuhause arbeiten und SML in der Version 110.0.7 verwenden, brauchen Sie anstelle der oben angegebenen Datei die Datei `3-6-110_0_7.sml`. Irgendwo zwischen Version 110.0.7 und der aktuellen Version 110.59 (die am CIP-Pool installiert ist) gab es eine Änderung in einer Signatur. Daran können Sie sehen, dass die Änderung von Signaturen viel Aufwand nach sich ziehen kann, denn deshalb müssen wir nun zwei Versionen zur Zeitmessung bereithalten.