

Informatik II, SoSe 2008

Übungsblatt 3

Abgabe: bis Mi 14.05.2008 7:30 Uhr

Besprechung 15.05.08 - 20.05.08

Aufgabe 3-1 SML

Rekursive Definition (wurzel)

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  $\varepsilon$ , 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 $ w^2 - x $	$< \varepsilon$ ?	Verbesserung $\frac{1}{2}(w + \frac{x}{w})$
1.0	1.0	nein	1.5
1.5	0.25	nein	1.4166666667
1.4166666667	0.00694444445389	nein	1.41421568627
1.41421568627	0.00000600729212685	ja, also Iterationswert	1.41421568627

In der WWW-Seite der Vorlesung finden Sie unter „Dateien“ 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.

**Hinweise:** 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. Wenn Sie mit anderen Werten von `epsilon` experimentieren, achten Sie bitte darauf, dass es bei der Abgabe als  $10^{-10}$  definiert ist.

Aufgabe 3-2 schriftlich bearbeiten (a) und SML (b)

Rekursive Definition (quadrat)

Hiasl Hirndübl grübelt über der folgenden Tabelle. Ihm fällt auf, dass die Zahl in der letzten Spalte jeweils die Summe der Zahlen in den beiden mittleren Spalten ist. Er meint, dass das kein Zufall ist, und verallgemeinert seine Beobachtung zu einer rekursiven Gleichung, über die er „Vermutung“ schreibt.

$n$	$n$ -te ungerade Zahl	$(n-1)^2$	$n^2$
0	—	—	0
1	1	0	1
2	3	1	4
3	5	4	9
4	7	9	16

$n$ -te ungerade Zahl =  $2n - 1$

Vermutung:

$$n^2 = \begin{cases} 0 & \text{falls } n = 0 \\ \dots & \text{falls } n > 0 \end{cases}$$

- a) Vervollständigen Sie die Vermutung und beweisen Sie sie. Hinweis: Sie können den Beweis machen ohne vollständige Induktion über  $n$  zu benutzen; es geht einfacher wenn Sie eine Fallunterscheidung machen wie in der Vermutung. Sie können für  $n^2$  zum Beispiel  $n \cdot 2$  schreiben.
- b) Hanni Hacker kommt dazu und sagt: „Wenn du jetzt noch  $2n$  durch  $n + n$  ausdrückst, kommt in deiner Gleichung gar keine Multiplikation mehr vor. Dann kann man damit eine Funktion `quadrat(n)` zum Quadrieren einer natürlichen Zahl rekursiv definieren, so dass nur Addition und Subtraktion in der Definition gebraucht werden, aber keine Multiplikation oder Division.“

Während Hiasl noch daran herumgrübelt, hat Hanni schon eine Datei `3-2b.sml` eingetippt, in der sie ihm die SML-Definition der Multiplikation und Division „abklemmt“. Diese Datei finden Sie in der WWW-Seite der Vorlesung unter „Dateien“, aber Sie brauchen nicht nachzuvollziehen, wie Hanni das programmiert hat.

Kopieren Sie sich die Datei in Ihr Unterverzeichnis für dieses Übungsblatt. Ergänzen Sie Ihre Kopie der Datei um eine SML-Definition von `quadrat(n)` gemäß der rekursiven Gleichung, und geben Sie die modifizierte Datei `3-2b.sml` als Lösung ab. Sie können voraussetzen, dass keine negativen Zahlen vorkommen. Es ist naheliegend, eine Hilfsfunktion zur Berechnung der  $n$ -ten ungeraden Zahl zu definieren, die dann aber ebenfalls nur Addition und Subtraktion zur Verfügung hat.

### Aufgabe 3-3 SML Anonyme Funktionen

Gegeben sind folgende Funktionen:

```
fun quadrat x = x*x;
fun fak n = if n = 0 then 1 else n * fak(n-1);
fun summe(x, y) = x + y;
```

Definieren Sie diese Funktionen als anonyme Funktionen, und binden Sie die anonyme Variante der Funktion `quadrat` an die Variable `quadrat2`, und ebenso für die anderen Funktionen. Definieren Sie für die Funktion `fak` ausserdem eine anonyme Variante `fak3` die Pattern Matching benutzt.

### Aufgabe 3-4 SML Rekursive Definition (produkt)

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  $\leq 0$  ist.

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

Die Datei `3-4a.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  $a \cdot b = a + a \cdot (b - 1)$ )

- 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 `3-4b.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.