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

München, 10.11.2006

Informatik 1
WS 2006/07

Übungsblatt 4: Rekursion, Substitutionsmodell, Auswertung von Sonderausdrücken

Besprechung: 20.11.–24.11.2006

Abgabe aller mit **Hausaufgabe** markierten Aufgaben bis Freitag, 17.11.2006, 18:00 Uhr

Aufgabe 4-1 *Rekursive Definition der Quadrat-Funktion* **(Hausaufgabe)**

Wenn Sie die folgende Tabelle betrachten, fällt Ihnen (hoffentlich) auf, dass die Zahl in der letzten Spalte jeweils die Summe der Zahlen in den beiden mittleren Spalten ist. Wir können vermuten, dass das kein Zufall ist, und diese „Vermutung" zu einer rekursiven Gleichung verallgemeinern.

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 (und zwar **nicht** durch vollständige Induktion über n; es geht viel einfacher mit einer Fallunterscheidung).

Schreiben Sie die Gleichung und den Beweis in eine Datei `4-1a.txt` und geben Sie diese Datei als Lösung ab. In dieser Textdatei können Sie für n^2 zum Beispiel `n^2` schreiben.

(b) In der *richtigen* Lösung zur ersten Teilaufgabe kommt als einzige Multiplikation $2n$ vor. Dies können wir auch durch $n+n$ ausdrücken, dann müssen wir gar keine Multiplikation mehr verwenden, um das Quadrat einer Zahl zu bestimmen. Wir können also 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.

In der Datei `4-1b.sml` auf der Vorlesungsseite ist die SML-Definition der Multiplikation und Division „abgeklemmt". 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 `4-1b.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 4-2 *Substitutionsmodell* **(Hausaufgabe)**

Kommen wir nun wieder zur „klassischen" `quadrat`-Funktion:

```
fun quadrat(z)             =  z * z;
fun summe_quadrate(x,y)    =  quadrat(x) + quadrat(y);
```

Geben Sie für den Ausdruck

`summe_quadrate(5-2, quadrat(3-1))`

die einzelnen Schritte bei der Auswertung an

(a) bei applikativer Auswertungsreihenfolge.

(b) bei normaler Auswertungsreihenfolge.

(c) bei verzögerter Auswertungsreihenfolge.

Schreiben Sie die Schritte für alle Teilaufgaben in eine Datei `4-2.txt` und geben Sie diese Datei als Lösung ab. In dieser Textdatei können Sie nicht zeichnen, aber Verbindungslinien etwa so darstellen:

```
   _____________________________
  /                             |
( ( 7 - 4 )  *  . )      +      .
   \___________|
```

wenn vom rechten Punkt auf `(7-4)` gezeigt werden soll und vom linken Punkt auf `7`

Achten Sie bitte darauf, keine Tabulatoren zu benutzen, da sich sonst die Darstellung Ihrer Lösung bei unterschiedlichen Editor-Einstellungen sehr stark von der gewollten Darstellung unterscheiden kann.

Aufgabe 4-3 *Auswertung von Sonderausdrücken* **(Hausaufgabe)**

Betrachten Sie folgende Funktionsdefinition in SML:

```
fun ganz(x)  =  x=0  orelse  ganz(abs(x)-1);
```

Diese Funktion bildet jeden `int`-Wert auf den Wahrheitswert `true` ab. Auch hierfür kann man also Rekursion verwenden. Ob das sinnvoll ist, interessiert uns nicht weiter. Aber wir können interessante Beobachtungen für das Auswertungsverhalten dieser Funktion anstellen.

(a) Beschreiben Sie mit dem Substitutionsmodell, wie der Ausdruck `ganz(~1)` in SML ausgewertet wird. Geben Sie genügend viele Zwischenschritte an, damit unmissverständlich klar wird, welche Teilausdrücke in welcher Reihenfolge ausgewertet werden und welchen Wert sie jeweils haben.

(b) Um den komischen Namen **`orelse`** nicht immer verwenden zu müssen, definieren wir uns eine zweite Funktion für die logische Operation $\vee$:

```
fun or(A1,A2)   =   A1  orelse  A2;
```

Nun können wir unsere Funktion `ganz` auch mit Hilfe unserer neuen Funktion `or` definieren:

```
fun ganz'(x)   =  or( x=0,  ganz'(abs(x)-1) );
```

Was ändert sich nun am Auswertungsverhalten der Funktion? Beschreiben Sie mit dem Substitutionsmodell, wie der Ausdruck `ganz'(~1)` in SML ausgewertet wird.

(c) Alternativ könnten wir mit Hilfe von Pattern-Matching die $\vee$-Funktion auch als Abbild der Wahrheitstafel implementieren – das entpräche dem ersten Ansatz in der Vorlesung:

```
fun or'(false, false)  =  false
  | or'(false, true )  =  true
  | or'(true,  false)  =  true
  | or'(true,  true )  =  true;
```

Definieren wir nun mit der Funktion `or'` unsere experimentelle Funktion:

```
fun ganz''(x) = or'( x=0,  ganz''(abs(x)-1) );
```

Bei applikativer Auswertungsreihenfolge (also in SML) wird sich am Auswertungsverhalten dieser Funktion nichts verändern. Warum nicht?

(d) Die *normale* Auswertungsreihenfolge ist für Funktionsdefinitionen mit Pattern Matching so definiert:

Wo im Muster der Funktionsdefinition eine Variable steht, wird so ausgewertet wie immer. Wo aber im Muster der Funktionsdefinition eine Konstante steht, wird von der normalen Auswertungsreihenfolge abgewichen und erst der entsprechende aktuelle Parameter ausgewertet, damit überhaupt entschieden werden kann, welche Gleichung der Definition verwendet werden kann.

Geben Sie eine Funktion `or''` mit Verwendung von pattern-Matching an, so dass die Funktion

```
fun ganz'''(x) = or''( x=0,  ganz'''(abs(x)-1) );
```

terminieren würde, falls SML in normaler Auswertungsreihenfolge auswerten würde. Geben Sie die einzelnen Schritte der Auswertung des Ausdrucks `ganz'''(~1)` an, wenn mit dieser Erweiterung der„normalen Auswertungsreihenfolge“ ausgewertet wird.

Geben Sie Ihre ausführlichen Auswertungen und Antworten für alle Teilaufgaben in einer Text-Datei `4-3.txt` ab.