

Programmierung und Modellierung, SS 2008
 Übungsblatt 4

Abgabe: Di 21.05.2008 nachts

Besprechung am Do-Di 23-28.05.2008

Aufgabe 4-1 Substitutionsmodell, verzögerte Auswertung

Text

```
fun q(z)      = z * z;
fun qsum(x,y) = q(x) + q(y);
```

Geben Sie für den Ausdruck `qsum(5-2, q(3-1))` die einzelnen Schritte bei der Auswertung an, wenn mit „verzögerter Auswertungsreihenfolge“ ausgewertet wird.

Schreiben Sie die Schritte in eine Datei `4-1.txt` und geben Sie diese Datei als Lösung ab. In dieser Textdatei können Sie nicht zeichnen, aber Verbindungslinien etwa so darstellen:

$$\begin{array}{c} \hline ((7 - 4) * .) + . \\ \hline \end{array}$$

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

Aufgabe 4-2 Applikative Auswertung, Sonderausdrücke

Text

a)

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

Diese Funktion bildet jeden `int`-Wert auf den Wahrheitswert `true` ab. Dass man sogar dazu Rekursion verwenden kann, hat Hiasl Hirndübl ausgetüftelt.

Beschreiben Sie in einer Datei `4-2a.txt` 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) Bartl Bastscho bastelt sich seine eigene Disjunktion, weil er sich den komischen Namen **orelse** so schlecht merken kann:

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

Er testet die Funktion erfolgreich mit Beispielen wie `or(5<3, 5>=3)` und ist gerade dabei, sie als Infixoperator zu definieren, als Vroni Frogstmi vorbeikommt und sagt: „Also die Funktion vom Hiasl würde damit überhaupt nicht terminieren. Du hast ja genau die Wahrheitstafel implementiert. In der Vorlesung haben wir aber gehört, dass **orelse**-Ausdrücke in SML nicht mit der Wahrheitstafel ausgewertet werden, sondern als Sonderausdrücke.“

Bartl erinnert sich nur undeutlich, aber er ändert seine Definition. Zum Testen baut er sein `or` in die Funktion von Hiasl ein. Er ist verblüfft, als er merkt, dass sie auch so nicht terminiert:

```

fun or(A1,A2)    =   A1 orelse A2;
fun ganz'(x)     =   or( x=0,  ganz'(abs(x)-1) );

```

Beschreiben Sie in einer Datei `4-2b.txt` mit dem Substitutionsmodell, wie der Ausdruck `ganz' (~1)` in SML ausgewertet wird.

Aufgabe 4-3 Normale Auswertung, Sonderausdrücke

Text

Haskell ist eine funktionale Programmiersprache mit verzögerter Auswertung, d.h., das Substitutionsmodell mit normaler Auswertungsreihenfolge ist korrekt für Haskell. Ansonsten ist Haskell SML sehr ähnlich; das Schlüsselwort **fun** fehlt, wie auch das Semikolon, Gleichheit ist `==`, negative Konstanten sind `-1`, `-2`, ... und die Wahrheitswerte werden großgeschrieben, also `True` und `False`. Betrachten Sie folgendes Haskell-Programm:

```

or'(True, _)    =   True
or'(False, b)   =   b

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

```

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 für den Ausdruck `ganz' (-1)` die einzelnen Schritte bei der Auswertung an, wenn mit dieser Erweiterung der „normalen Auswertungsreihenfolge“ ausgewertet wird.

Aufgabe 4-4 Referenzen

SML

- a) Zu jeder Kennung gibt es ein Drucker-Kontingent, das angibt, wie viele Seiten mit dieser Kennung noch gedruckt werden können. Diese Kontingente werden *Quotas* genannt. Die Datei `4-4a.sml` enthält einen Rahmen für die Verwaltung solcher Quotas mit SML. Vervollständigen Sie die Definitionen der folgenden Funktionen und geben Sie die modifizierte Datei ab.

```
new_quota(initial_pages : int)
```

Liefert bei jedem Aufruf ein neues Quota, das so viele Seiten zur Verfügung stellt, wie der Wert von `initial_pages` angibt.

```
reduce(quota : int ref, pages : int) : unit
```

`quota` ist ein Quota das zum Beispiel mit `new_quota(200)` erzeugt sein kann.

`pages` gibt an, wie viele Seiten von diesem Quota gedruckt werden sollen.

Wenn das Quota gar nicht mehr genügend Seiten zu drucken erlaubt,

wird mit **raise** `quota_too_low` eine Ausnahme ausgelöst.

Andernfalls wird das Quota um die Anzahl der Seiten verkleinert.

```
pages(quota : int ref) : int
```

`quota` ist ein Quota, das zum Beispiel mit `new_quota(200)` erzeugt sein kann.

Liefert die Anzahl der Seiten, die dieses Quota noch zu drucken erlaubt.

Gibt man folgendes nacheinander in SML ein, hat der letzte Ausdruck den Wert 42:

```
val q = new_quota(200);  reduce(q,100);  reduce(q,58);  pages(q);
```

- b) In einer neuen SML-Sitzung werden hintereinander drei Eingaben gemacht:

1. die Datei der vorigen Teilaufgabe laden

2. **val** `q = new_quota(200);`

3. Boole'scher Ausdruck der Gestalt `TEILAUSDRUCK = TEILAUSDRUCK`; wo links und rechts vom Gleichheitszeichen syntaktisch der gleiche Teilausdruck steht.

Geben Sie in einer Datei `4-4b.txt` einen solchen Boole'schen Ausdruck an, der den Wert `false` hat. Der `TEILAUSDRUCK` kann mit den Funktionsnamen der vorigen Teilaufgabe sowie `q` und natürlichen Zahlen gebildet werden.