TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Einführung in die Informatik 2
Prof. Dr. Helmut Seidl, T. M. Gawlitza, S. Pott, M. Schwarz

WS 2008/09
Übungsblatt 11
23.12.2008

Abgabe: 13.01.2009 (vor der Vorlesung)

Aufgabe 11.1 (H) In großen Schritten zum Ziel

Gegeben seien folgende MiniOCaml-Definitionen:

```
let f = (fun x -> (fun y -> 2 * x + y))

let g = f 7

let twice = fun f1 -> fun a -> f1 (f1 a)

let rec unzip =
  fun l ->
    match l with
        [] -> ([],[])
      | (x1,x2)::xs ->
        (
          match unzip xs with
            (xs1,xs2) -> ((x1::xs1),(x2::xs2))
        )
```

Konstruieren Sie die Beweise für folgende Aussagen:

a) twice g 7 $\Rightarrow$ 35

b) unzip [(1,2)] $\Rightarrow$ ([1],[2])

Aufgabe 11.2 (H) Abgeleitete Regeln

Es wird die folgende abgeleitete Regel betrachtet.

$$\frac{\mathtt{e}_0 = \mathtt{fun\ x} \to \mathtt{e} \quad \mathtt{e}_1 \text{ terminiert}}{\mathtt{e}_0\ \mathtt{e}_1 = \mathtt{e}[\mathtt{e}_1/\mathtt{x}]}$$

a) Zeigen Sie anhand eines Gegenbeispiels, dass auf die Voraussetzung „$\mathtt{e}_1$ terminiert“ nicht verzichtet werden kann.

b) Zeigen Sie die Gültigkeit obiger Regel.

Aufgabe 11.3 (P) Neues von `fact`, `map` und `comp`

a) Es seien folgende, bekannte Definitionen gegeben:

```
let rec fact = fun n ->
  match n with
    0 -> 1
  | n -> n * fact (n-1)

let rec fact_aux = fun x n ->
  match n with
    0 -> x
  | n -> fact_aux (n*x) (n-1)

let fact_iter = fact_aux 1
```

Zeigen Sie, dass unter der Voraussetzung, dass alle Aufrufe terminieren,

$$\texttt{fact\_iter}\ \texttt{n} = \texttt{fact}\ \texttt{n}$$

für alle nicht-negativen ganzen Zahlen $\texttt{n} \in \mathbf{N}_0$ gilt.

b) Es seien folgende, bekannte Definitionen gegeben:

```
let comp = fun f g x -> f (g x)

let rec map = fun op l ->
  match l with
    [] -> []
  | x::xs -> op x :: map op l
```

Zeigen Sie, dass unter der Voraussetzung, dass alle Aufrufe terminieren,

$$\texttt{map}\ (\texttt{comp}\ \texttt{f}\ \texttt{g}) = \texttt{comp}\ (\texttt{map}\ \texttt{f})\ (\texttt{map}\ \texttt{g})$$

für alle `f` und g gilt.

Big-Step Operationelle Semantik

Axiome: $v \Rightarrow v$ für jeden Wert v

Tupel:
$$\frac{e_1 \Rightarrow v_1 \quad \ldots \quad e_k \Rightarrow v_k}{(e_1, \ldots, e_k) \Rightarrow (v_1, \ldots, v_k)} \; (T)$$

Listen:
$$\frac{e_1 \Rightarrow v_1 \quad e_2 \Rightarrow v_2}{e_1 \texttt{::} e_2 \Rightarrow v_1 \texttt{::} v_2} \; (L)$$

Globale Definitionen:
$$\frac{f = e \quad e \Rightarrow v}{f \Rightarrow v} \; (GD)$$

Lokale Definitionen:
$$\frac{e_1 \Rightarrow v_1 \quad e_0[v_1/x] \Rightarrow v_0}{\texttt{let}\ x = e_1\ \texttt{in}\ e_0 \Rightarrow v_0} \; (LD)$$

Funktionsaufrufe:
$$\frac{e_1 \Rightarrow \texttt{fun}\ x \texttt{->} e_0 \quad e_2 \Rightarrow v_2 \quad e_0[v_2/x] \Rightarrow v_0}{e_1\ e_2 \Rightarrow v_0} \; (App)$$

Pattern Matching:
$$\frac{e_0 \Rightarrow v' \equiv p_i[v_1/x_1, \ldots, v_k/x_k] \quad e_i[v_1/x_1, \ldots, v_k/x_k] \Rightarrow v}{\texttt{match}\ e_0\ \texttt{with}\ p_1 \texttt{->} e_1\ \texttt{|}\ \ldots\ \texttt{|}\ p_m \texttt{->} e_m \Rightarrow v} \; (PM)$$

— sofern v' auf keines der Muster $p_1, \ldots, p_{i-1}$ passt

Eingebaute Operatoren:
$$\frac{e_1 \Rightarrow v_1 \quad e_2 \Rightarrow v_2 \quad v_1\ \mathsf{op}\ v_2 \Rightarrow v}{e_1\ \mathsf{op}\ e_2 \Rightarrow v} \; (Op)$$

— Unäre Operatoren werden analog behandelt.

Substitutionslemma

$$\frac{e_1 = e_2}{e[e_1/x] = e[e_2/x]}$$