*Technische Universität München*
*Fakultät für Informatik* *Peter Ziewer*
*Prof. Dr. H. Seidl*

# Abstrakte Maschinen

*Sommersemester 2006*

*7. Übungsblatt* *Abgabetermin: Do, 29. Juni 2006 in der Vorlesung*

Aufgabe 1: *Referenzen* *20 Punkte*

Referenzen in funktionalen Sprachen entsprechen den Variablen (und Zeigern) in imperativen Sprachen. Betrachten Sie folgendes Beispiel:

```
letrec
    x = 1;
    f = fn y => if y <= 0 then x
                else let x = x * 2 in f (y-1)
in f 8
```

`x = x * 2` erzeugt ein *neues* `x`, das nur innerhalb der Funktion `f` sichtbar ist. Das Ergebnis wird also immer 1 sein.

Betrachten Sie nun das Programm mit Referenzen:

```
letrec
    x = ref 1;
    f = fn y => if y <= 0 then !x
                else let z = (x := !x * 2) in f (y-1)
in f 8
```

`x = ref 1` weist der Variablen `x` eine Referenz auf den Wert 1 zu. Durch die Zuweisung `x := !x * 2` wird der Wert dieser Referenz-Variablen `x` verändert. Es wird jedoch *keine* neue Variable erzeugt. `!x` liefert den Wert der Referenz. Das Ergebnis lautet 256 ($= 2^8$).

Zur Implementierung führen wir Referenz-Objekte als neuen Datentyp des Heaps ein. Referenz-Objekte bestehen aus dem Tag $R$ und einem Verweis (auf einen Wert).

Geben Sie Codeerzeugungsfunktionen ($\text{Code}_V$) für folgende Ausdrücke an:

a) `ref` $e$
   erzeugt ein neues Referenz-Objekt für den Ausdruck $e$ und legt einen Verweis darauf auf den Stack.

b) `!`$e$
   liefert den Wert der durch den Ausdruck $e$ bestimmten Referenz.

c) $e_1$ `:=` $e_2$
   weist der durch $e_1$ bestimmten Referenz den Wert des Ausdrucks $e_2$ zu und legt diesen Wert auf den Stack.

Definieren Sie gegebenenfalls neue Instruktionen (z.B. `mkref` oder `getref`).

(d) Übersetzen Sie das zweite Beispiel von oben.