Compilerbau

http://proglang.informatik.uni-freiburg.de/teaching/compilerbau/2006ws/

Übungsblatt 10

17.1.2006

Aufgabe 1 (Anwendung von Lambda–Lifting; 4 Punkte)
Übersetzen sie mittels Lambda–Lifting die folgenden zwei OCaml Programme in äquivalente OCaml Programme welche `let rec` Bindungsgruppen nur auf der obersten Ebene haben.

(i)
```
let a = 5 in
    let rec f x = if x <= 0 then a else x * f (x - 1) in
      f a;;
```

(ii)
```
let a = 5 in
let rec f x = if x <= 0 then a else x * f (x - 1) in
  f (let rec g x y = if x <= 0 then y else a + h x
         and h x   = g (x - 1) a
     in h 11)
```

Aufgabe 2 (Implementierung von Lambda–Lifting; 6 Punkte)
Seien folgende OCaml Typdefinitionen gegeben: (Der Code ist auch auf der Homepage erhältlich.)

```
type id = string

type 'e equation = Equation of id      (* function name *)
                             * id list (* arguments *)
                             * 'e      (* body *)
(* Source language *)

type exp = Var of id
         | App of exp * exp                    (* function application *)
         | Letvar of id * exp * exp            (* let x = 5 in e *)
         | Letrec of (exp equation) list * exp (* letrec f_1 x_1 ... x_n1 = e_m
                                                           ...
                                                           f_m x_1 ... x_nm = e_m
                                                     in e *)


(* Target language *)

type scheme = Scheme of (exp' equation) list * exp'
```

```
and exp' = Var' of id
         | App' of exp' * exp'
         | Letvar' of id * exp' * exp'
```

Implementieren sie Lambda–Lifting, um einen Ausdruck vom Typ `exp` in einen Ausdruck vom Typ `exp'` zu übersetzen. Sie können dabei annehmen, dass kein Bezeichner im Programm mehrfach gebunden wird und dass die mit `Letrec` ausgedrückten Bindungsgruppen minimal sind, d.h. alle Funktionen der Bindungsgruppe sind wirklich verschränkt rekursiv.

Der Code soll dabei in eine Datei namens `lambda_lift.ml` geschrieben werden. Einstiegspunkt soll eine Funktion `lift` vom Typ `exp -> exp'` sein.

Aufgabe 3 (A–Normal Form; 4 Punkte)
Transformieren sie das Programm aus Aufgabe 1(i) nach A–Normal Form.

Abgabe: 24.1.2007
Die Abgabe erfolgt bis zu Beginn der Übungsstunde. Einreichungen, die nicht den Abgabemodalitäten entsprechen, werden abgelehnt. Für Plagiate werden keine Punkte vergeben.

Abgabemodalitäten:

- Code muss per Email an die Adresse `wehr@informatik.uni-freiburg.de` geschickt werden. Zu diesem Blatt ist kein Papierabgabe nötig.
- Der Code muss in einem Archiv mit dem Namen `vorname_nachname.tar.gz` oder `vorname_nachname.zip` an die Email angehängt werden.
- Entpacken des Archivs muss ein einzelnes Verzeichnis mit dem Namen `vorname_nachname` liefern.
- Innerhalb dieses Verzeichnisses muss die Datei `lambda_lift.ml` (für Aufgabe 2) enthalten sein.
- Alle Dateien müssen compilierbar sein; Teile einer Datei, welche nicht vom Compiler akzeptiert werden, müssen auskommentiert sein. Abgaben, die der Compiler nicht akzeptiert, werden nicht bewertet.
- Wenn eine Aufgabe von der Lösung ein gewisses Format verlangt (wie etwa einen festen Namen oder Typ für eine Funktion, eine vorgegebene Signatur für ein Modul etc.), so ist dieses Format zwingend einzuhalten.