

---

**Compilerbau**


---

Blatt 8

Abgabe: 12.12.2001

- [10 Punkte] Geben Sie eine Definition für einen Rechts-nach-Links-Call-by-Name-Lambda-Kalkül an und untersuchen Sie diesen. Ist er noch in der Lage, mit Sicherheit schwache Kopf-Normalformen zu finden? Tun Sie das gleiche für einen Rechts-nach-Links-Call-by-Value-Lambda-Kalkül. Wie vergleicht dieser sich mit seinem Links-Nach-Rechts-Gegenstück?
- [5 Punkte] Implementieren Sie ein Lösungsverfahren für reine Vereinigungsprobleme. Gegeben sei also eine Relation  $R \subseteq A \times A$  auf einer Menge  $A$  und eine mengenwertige Funktion  $g : A \rightarrow M$ . Gesucht ist eine Funktion  $f : A \rightarrow M$ , welche folgende Gleichung löst:

$$f(a) = g(a) \cup \bigcup \{f(b) \mid aRb\}$$

- [15 Punkte] Implementieren Sie Lambda-Lifting als Abbildung

```
caml_to_lambda : Camlsyn.program -> Lambda.scheme
```

Die Datentypen für Lambda-Ausdrücke sind dabei im Modul `Lambda` wie folgt definiert:

```
type const =
  CInt of int
  | CString of string
  | CChar of char

type exp =
  Const of const
  | Ident of string
  | Builtin of string
  | Lambda of string * exp
  | Apply of exp * exp
  | Let of string * exp * exp
  | If of exp * exp * exp

type equation = string * exp

type scheme = equation list * exp
```

Wählen Sie für die Transformation den Aufruf einer Funktion `main : unit -> unit` als Eintrittspunkt. Sie können dabei annehmen, daß in der abstrakten Syntax alle Bezeichner wie in Aufgabe 3 auf Blatt 7 umbenannt worden sind. Ignorieren Sie dabei (für den Moment) `raise` und `try`.

Hinweis: Zur Implementierung des Lambda-Liftings benötigen Sie den Algorithmus für reine Vereinigungsprobleme aus Aufgabe 2. Sollten Sie diese Aufgabe nicht oder erfolglos bearbeitet haben, wenden Sie sich an Martin Gasbichler.