Institut für Informatik
und Praktische Mathematik
der Universität Kiel

Kiel, 16.12.1999

# Übungen zu Informatik I
# WS 1999/2000
## Serie 7

**Aufgabe T9** *(nicht für Physiker)* (6 Punkte)

Spezifizieren Sie in der in der Vorlesung (Abschnitt 2.3) eingeführten Sprache FL eine rekursive Variante der $\alpha$-Konversions-Funktion, die in einer Abstraktion der allgemeinen Form $\lambda z_1....\lambda z_n.e_b$ die gebundenen Variablen $z_1, ..., z_n$ ($n$ beliebig) in der Reihenfolge von außen nach innen (d.h. von $z_1$ nach $z_n$) in $v$ umbenennt. Sei $f$ der Bezeichner für diese Funktion, dann ist deren Anwendung auf die obige Abstraktion wie folgt definiert:

$$(f \quad \lambda z_1.\lambda z_2....\lambda z_n.e_b) \rightsquigarrow \lambda v.(f \quad \lambda z_2....\lambda z_n.e_b[z_1 \leftarrow v]) \quad .$$

Der rekursive Vortrieb von $f$ durch die Sequenz von Binde-Operatoren terminiert mit

$$\lambda v.\lambda v....\lambda v.(f \quad e_b[\ldots]) \rightsquigarrow \lambda v.\lambda v....\lambda v.e_b[\ldots] \quad .$$

Als Terminierungsbedingung verwenden Sie bitte eine Testfunktion `is_abstr` mit

$$(\texttt{is\_abstr} \quad e) = \begin{cases} true & \text{falls } e =_s \lambda w.e_b \\ false & \text{sonst} \end{cases} \quad .$$

Demonstrieren Sie durch schrittweise $\alpha$-Konversion des Terms

$$\lambda x.\lambda y.\lambda z.(x \quad (y \quad z)) \quad ,$$

dass dieser sich durch die von Ihnen spezifizierte Funktion tatsächlich in

$$\lambda v.\lambda v.\lambda v.(//v \quad (/v \quad v))$$

transformieren läßt.

**Anmerkung:** Leider stehen in KiR weder die Funktion `is_abstr` noch eine andere Möglichkeit, eine Abstraktion von anderen Termen per Wahrheitswert unterscheidbar zu machen, zur Verfügung. Deshalb hat es keinen Sinn, einen Versuch zu unternehmen, dieses Problem mit Hilfe des Reduktionssystems zu lösen.

**Aufgabe P11** (14 Punkte)

Pattern-Matches können in KiR außer über die in der Vorlesung eingeführten und in den vorangegangenen Übungen benutzten Listen-Patterns auch mit Hilfe sog. Konstruktor-Patterns, wie sie auf den Seiten 59, 60 des KiR-Manuals eingeführt und auf den nachfolgenden Seiten anhand eines Beispiels erläutert sind, spezifiziert werden.

Diese Konstruktor-Patterns ermöglichen es, sog. Meta-Sprachen in KiR zu spezifizieren und mit Hilfe von Pattern-Matches zu interpretieren. So lassen sich u.a. die Terme des reinen $\lambda$-Kalküls als Konstruktor-Terme wie folgt darstellen:

$$term \quad \Rightarrow \quad \texttt{Var\{ } u \texttt{ \}} \;\mid\; \texttt{Lam\{ } u \texttt{ , } term \texttt{ \}} \;\mid\; \texttt{Ap\{ } term \texttt{ , } term \texttt{ \}}$$

wobei $u$ bzw. *term* Platzhaltervariablen für konkrete, in dieser Meta-Sprache zu spezifizierende Variablen bzw. Terme sind. Konkrete Variablen müssen als Strings in der Form

$$u \quad \Rightarrow \quad \texttt{'x`} \;\mid\; \texttt{'y`} \;\mid\; \texttt{'www`} \;\mid\; \ldots$$

angegeben werden.

(1) Spezifizieren Sie zunächst eine (rekursive) KiR-Funktion

```
Subst [ U, BodyTerm, ArgTerm ] = ...
```

die alle Vorkommen von `U` in dem für `BodyTerm` eingesetzten Term durch den für `ArgTerm` eingesetzen Term *naiv* (d.h. ohne Rücksicht auf mögliche Namenskonflikte) substituiert.

Verwenden Sie hierfür entsprechende, über Konstruktor-Patterns zu spezifizierende Regeln, die die Funktion `Subst` rekursiv in komplexe Terme hineintreiben und an atomaren Termen die Substitution vornehmen.

(2) Spezifizieren Sie in ähnlicher Weise eine KiR-Funktion

```
Eval [ Term ] = ...
```

die entsprechend der im Abschnitt 2.3 der Vorlesung angegebenen Definition einen für den Parameter `Term` einzugebenen Term unter Zuhilfenahme der Funktion `Subst` auswertet.

(3) Testen Sie ihr Programm anhand der folgenden (vorher in Konstruktor-Syntax umzuwandelnden) $\lambda$-Terme:

(a) $(\lambda u.\lambda v.(u \;\; v) \;\; \lambda x.x)$

(b) $((\lambda u.\lambda v.(u \;\; v) \;\; \lambda x.\lambda y.x) \;\; z)$

(c) $(((\lambda u.\lambda v.\lambda w.((u \;\; v) \;\; (u \;\; w)) \;\; \lambda x.x) \;\; \lambda y.\lambda z.y) \;\; s)$

**Abgabe:** Bis Do., 13.01.2000, 11 Uhr s.t. im Schrein (Institut für Informatik, Haus I, Erdgeschoß).