Übung zur Vorlesung

Automatisierte Logik und Programmierung I

Prof. Chr. Kreitz

Universität Potsdam, Theoretische Informatik — Wintersemester 2008/09

Blatt 3 — Abgabetermin: 26.11.08 nach der Übung

Das dritte Übungsblatt soll dazu dienen, mit dem λ–Kalkül vertraut zu werden. Dazu soll ein Einstieg über das Anwenden von Definitionen für Standard–Erweiterungen versucht werden.

Aufgabe 3.1 (λ–Kalkül I: Projektionen)

Zeigen Sie, daß die Operatoren $pair.\mathbf{1} \equiv pair\ (\lambda x.\lambda y.x)$ und $pair.\mathbf{2} \equiv pair\ (\lambda x.\lambda y.y)$ tatsächlich die Projektionen eines Paares $pair = \langle s,t\rangle \equiv \lambda p.\ p\ s\ t$ berechnen.

Aufgabe 3.2 (λ–Kalkül II: Bool'sche Algebra)

Definieren Sie mit Hilfe der Bool'schen Operatoren **T**, **F** und **if** b **then** s **else** t die folgenden Operatoren:

3.2–a "**and**", die logische Konjunktion

3.2–b "**or**", die logische Disjunktion

3.2–c "**neg**", die logische Negation

3.2–d "**imp**", die logische Implikation

Überlegen Sie sich, wie man die Korrektheit dieser Definitionen nachweisen könnte und führen Sie für diesen Nachweis für einen der Operatoren mit Hilfe der Beweisregeln für den λ-Kalkül.

Aufgabe 3.3 (λ–Kalkül III: Church Numerals)

Die Darstellung der natürlichen Zahlen wurde mit Hilfe der *Church Numerals* ($\overline{n} \equiv \lambda f.\lambda x.\ f^n\ x$) beschrieben. Zeigen Sie

3.3–a $\mathbf{zero} \equiv \lambda n.\ n\ (\lambda n.\mathbf{F})\ \mathbf{T}$ beschreibt einen Test auf Null, d.h.: $\mathbf{zero}\ \overline{n} = \begin{cases} \mathbf{T} & \text{wenn } n = 0, \\ \mathbf{F} & \text{wenn } n > 0 \end{cases}$

3.3–b $\mathbf{mul} \equiv \lambda m.\lambda n.\lambda f.\lambda x.\ m\ (n\ f)\ x$ repräsentiert die Multiplikation, d.h.: $\mathbf{mul}\ \overline{m}\ \overline{n} = \overline{m \cdot n}$

3.3–c $\mathbf{p} \equiv \lambda n.\ (n\ (\lambda z.\ \langle \mathbf{s}, \mathbf{let}\ \langle f,x\rangle = z\ \mathbf{in}\ f\ x\rangle)\ \langle \lambda z.\overline{0}, \overline{0}\rangle).\mathbf{2}$
repräsentiert die Vorgängerfunktion, d.h.: $\mathbf{p}\ \overline{n} = \overline{n-1}$. (sehr aufwendig)

Aufgabe 3.4 (λ–Kalkül IV: Ganzzahlfunktionen)

3.4–a Geben Sie einen λ-Term **subtract** an mit der Eigenschaft $\mathbf{subtract}\ \overline{m}\ \overline{n} = \overline{m-n}$

3.4–b Beschreiben Sie einen λ-Term **less_or_equal** mit $\mathbf{less\_or\_equal}\ \overline{m}\ \overline{n} = \begin{cases} \mathbf{T} & \text{falls } m \le n \\ \mathbf{F} & \text{sonst} \end{cases}$

3.4–c Geben Sie einen λ-Term **max** an mit der Eigenschaft $\mathbf{max}\ \overline{m}\ \overline{n} = \begin{cases} \overline{n} & \text{falls } m \le n \\ \overline{m} & \text{sonst} \end{cases}$

Zeigen Sie beispielhaft, daß Ihre Terme korrekt sind.

Lösung 3.1 *Ziel dieser Aufgabe ist es, ein bißchen mit λ–Termen herumzuspielen. Als Testfeld dienen dazu die ziemlich einfachen Definitionen für die Projektionen von Paarungen:*

$$
\begin{array}{lcl}
\langle u,v\rangle.\mathbf{1} & \equiv & \langle u,v\rangle\ (\lambda x.\lambda y.\ x) \equiv (\lambda p.\ p\,u\,v)(\lambda x.\lambda y.\ x) \\
& \longrightarrow & (\lambda x.\lambda y.\ x)\,u\,v \\
& \longrightarrow & u \\
\langle u,v\rangle.\mathbf{2} & \equiv & \langle u,v\rangle\ (\lambda x.\lambda y.\ y) \equiv (\lambda p.\ p\,u\,v)(\lambda x.\lambda y.\ y) \\
& \longrightarrow & (\lambda x.\lambda y.\ y)\,u\,v \\
& \longrightarrow & v
\end{array}
$$

Lösung 3.2 *Ziel dieser Aufgabe ist es, dem oben beschriebenen Gesamtziel dieses Übungsblattes in Form einer "Vorspeise" näher zu kommen. Wir üben uns zunächst an den einfacheren Dingen des "Lebens" — Bool'sche Algebra. . .*

3.2–a *"***and***", die logische Konjunktion:*

Eine kurze Überlegung führt zu folgender Fallunterscheidung:

1. *Ist A wahr, so ist $A \wedge B$ wahr, genau dann wenn B wahr ist.*
2. *Ist A falsch, so ist $A \wedge B$ sowieso falsch.*

Damit können wir **and** *wie folgt definieren:*

$$
\begin{array}{lcl}
\textbf{and} & \equiv & \lambda a.\lambda b.\textbf{if}\ a\ \textbf{then}\ b\ \textbf{else}\ \textbf{F} \\
& \equiv & \lambda a.\lambda b.\textbf{cond}(a;b;\textbf{F}) \\
& \equiv & \lambda a.\lambda b.\ a\ b\ \textbf{F}
\end{array}
$$

*Wir zeigen zunächst den Beweis im Sequenzenkalkül für die Gleichheit von λ–Termen mit dem Eingabefall ≫**T T**≪:*

```
⊢ (λa,b.a b (λx,y.y)) (λx,y.x)(λx,y.x) = λx,y.x
                                    BY transitivity '(λb.(λx,y.x) b (λx,y.y))(λx,y.x)'
|\
| ⊢ (λa,b.a b (λx,y.y))(λx,y.x)(λx,y.x) = (λb.(λx,y.x) b (λx,y.y))(λx,y.x)
|                                                                       BY applyEq
| |\
| | ⊢ (λa,b.a b (λx,y.y))(λx,y.x) = λb.(λx,y.x) b (λx,y.y)            BY reduction
| | \
| |  ⊢ λb.(λx,y.x) b (λx,y.y) = λb.(λx,y.x) b (λx,y.y)              BY reflexivity
|  \
|   ⊢ λx,y.x = λx,y.x                                                 BY reflexivity
\
 ⊢ (λb.(λx,y.x) b (λx,y.y))(λx,y.x) = λx,y.x                         BY reduction
 \
  ⊢ (λx,y.x)(λx,y.x)(λx,y.y) = λx,y.x     BY transitivity '(λy,x,y.x)(λx,y.y)'
  |\
  | ⊢ (λx,y.x)(λx,y.x)(λx,y.y) = (λy,x,y.x)(λx,y.y)                  BY applyEq
  | |\
  | | ⊢ (λx,y.x)(λx,y.x) = λy,x,y.x                                  BY reduction
  | | \
  | |  ⊢ λy,x,y.x = λy,x,y.x                                         BY reflexivity
  |  \
  |   ⊢ λx,y.y = λx,y.y                                               BY reflexivity
   \
    ⊢ (λy,x,y.x)(λx,y.y) = λx,y.x                                     BY reduction
     \
      ⊢ λx,y.x = λx,y.x                                              BY reflexivity
```

Als nächstes folgt der Fall ≫**F T**≪*:*

```
⊢ (λa,b.a b (λx,y.y))(λx,y.y)(λx,y.x) = λx,y.y
                                   BY transitivity '(λb.(λx,y.y) b (λx,y.y))(λx,y.x)'
|\
| ⊢ (λa,b.a b (λx,y.y))(λx,y.y)(λx,y.x) = (λb.(λx,y.y) b (λx,y.y))(λx,y.x)
|                                                                       BY applyEq
| |\
| | ⊢ (λa,b.a b (λx,y.y))(λx,y.y) = λb.(λx,y.y) b (λx,y.y)          BY reduction
| |  \
| |   ⊢ λb.(λx,y.y) b (λx,y.y) = λb.(λx,y.y) b (λx,y.y)          BY reflexivity
|  \
|   ⊢ λx,y.x = λx,y.x                                              BY reflexivity
 \
  ⊢ (λb.(λx,y.y) b (λx,y.y))(λx,y.x) = λx,y.y                       BY reduction
   \
    ⊢ (λx,y.y)(λx,y.x)(λx,y.y) = λx,y.y         BY transitivity '(λy.y)(λx,y.y)'
    |\
    | ⊢ (λx,y.y)(λx,y.x)(λx,y.y) = (λy.y)(λx,y.y)                        BY applyEq
    | |\
    | | ⊢ (λx,y.y)(λx,y.x) = λy.y|                                     BY reduction
    | |  \
    | |   ⊢ λy.y = λy.y                                              BY reflexivity
    |  \
    |   ⊢ λx,y.y = λx,y.y                                            BY reflexivity
     \
      ⊢ (λy.y)(λx,y.y) = λx,y.y                                        BY reduction
       \
        ⊢ λx,y.y = λx,y.y                                            BY reflexivity
```

Alle anderen Fälle sind derart analog, daß wir sie uns ersparen wollen. . .

3.2–b *"***or***", die logische Disjunktion:*

Analog zu den obigen Überlegungen kommen wir zu folgender Definition für **or***:*

$$\begin{aligned}\textbf{or} &\equiv \lambda\texttt{a.}\lambda\texttt{b.}\textbf{if}\ \texttt{a}\ \textbf{then}\ \textbf{T}\ \textbf{else}\ \texttt{b}\\ &\equiv \lambda\texttt{a.}\lambda\texttt{b.}\textbf{cond}(\texttt{a;}\textbf{T}\texttt{;b})\\ &\equiv \lambda\texttt{a.}\lambda\texttt{b.}\ \texttt{a}\ \textbf{T}\ \texttt{b}\end{aligned}$$

Auch hier gilt wiederum, daß die Sequenzenbeweise extrem analog zu den oben vorgeführten sind, so daß wir sie geflissentlich weglassen wollen.

3.2–c *"***neg***", die logische Negation:*

$$\begin{aligned}\textbf{neg} &\equiv \lambda\texttt{a.}\textbf{if}\ \texttt{a}\ \textbf{then}\ \textbf{F}\ \textbf{else}\ \textbf{T}\\ &\equiv \lambda\texttt{a.}\textbf{cond}(\texttt{a;}\textbf{F}\texttt{;}\textbf{T})\\ &\equiv \lambda\texttt{a.}\ \texttt{a}\ \textbf{F}\ \textbf{T}\end{aligned}$$

Sequenzenbeweise s. o.

3.2–d *"***imp***", die logische Implikation:*

$$\begin{aligned}\textbf{imp} &\equiv \lambda\texttt{a.}\lambda\texttt{b.}\textbf{if}\ \texttt{a}\ \textbf{then}\ \texttt{b}\ \textbf{else}\ \textbf{T}\\ &\equiv \lambda\texttt{a.}\lambda\texttt{b.}\textbf{cond}(\texttt{a;b;}\textbf{T})\\ &\equiv \lambda\texttt{a.}\lambda\texttt{b.}\ \texttt{a}\ \texttt{b}\ \textbf{T}\end{aligned}$$

Sequenzenbeweise s. o.

Lösung 3.3 *Ziel dieser Aufgabe ist es, den Umgang mit dem λ–Kalkül weiter zu vertiefen. Dabei soll wiederum eine der Standard–Erweiterungen als Übungsfeld herhalten. In diesem Falle sind es die — etwas unübersichtlichen —* Church Numerals*:*

3.3–a **zero** $\overline{n} =$ ***T***, *falls* $n = 0$, ***F*** *sonst:*

$$
\begin{array}{rcl}
\textbf{zero}\ \overline{0} & \equiv & \textbf{zero}\ (\lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{x}) \\
& \equiv & (\lambda \mathtt{n}.\ \mathtt{n}\ (\lambda \mathtt{n}.\ \mathbf{F})\ \mathbf{T})\ (\lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{x}) \\
& \longrightarrow & (\lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{x})\ (\lambda \mathtt{n}.\ \mathbf{F})\ \mathbf{T} \\
& \longrightarrow & \mathbf{T}
\end{array}
$$

$$
\begin{array}{rcl}
\textbf{zero}\ \overline{n+1} & \equiv & \textbf{zero}\ (\lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{f}^{n+1}\ \mathtt{x}) \\
& \equiv & (\lambda \mathtt{n}.\ \mathtt{n}\ (\lambda \mathtt{n}.\ \mathbf{F})\ \mathbf{T})\ (\lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{f}^{n+1}\ \mathtt{x}) \\
& \longrightarrow & (\lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{f}^{n+1}\ \mathtt{x})\ (\lambda \mathtt{n}.\ \mathbf{F})\ \mathbf{T} \\
& \longrightarrow & (\lambda \mathtt{x}.(\lambda \mathtt{n}.\ \mathbf{F})^{n+1}\ \mathtt{x})\ \mathbf{T} \\
& \longrightarrow & (\lambda \mathtt{n}.\ \mathbf{F})^{n+1}\ \mathbf{T} \\
& \equiv & (\lambda \mathtt{n}.\ \mathbf{F})^{n}\ ((\lambda \mathtt{n}.\ \mathbf{F})\ \mathbf{T}) \\
& \longrightarrow & (\lambda \mathtt{n}.\ \mathbf{F})^{n}\ \mathbf{F} \\
& \equiv & (\lambda \mathtt{n}.\ \mathbf{F})^{n-1}\ ((\lambda \mathtt{n}.\ \mathbf{F})\ \mathbf{F}) \\
& \longrightarrow & (\lambda \mathtt{n}.\ \mathbf{F})^{n-1}\ \mathbf{F} \\
& \vdots & \vdots \\
& \longrightarrow & \mathbf{F}
\end{array}
$$

3.3–b **mul** $\overline{m}\ \overline{n} = \overline{m \cdot n}$*:*

$$
\begin{array}{rcl}
\textbf{mul}\ \overline{m}\ \overline{n} & \equiv & (\lambda \mathtt{m}.\lambda \mathtt{n}.\lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{m}\ (\mathtt{n}\ \mathtt{f})\ \mathtt{x})\ \overline{m}\ \overline{n} \\
& \longrightarrow & \lambda \mathtt{f}.\lambda \mathtt{x}.\ \overline{m}\ (\overline{n}\ \mathtt{f})\ \mathtt{x} \\
& \equiv & \lambda \mathtt{f}.\lambda \mathtt{x}.\ (\lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{f}^{m}\ \mathtt{x})\ (\overline{n}\ \mathtt{f})\ \mathtt{x} \\
& \longrightarrow & \lambda \mathtt{f}.\lambda \mathtt{x}.\ (\overline{n}\ \mathtt{f})^{m}\ \mathtt{x} \\
& \equiv & \lambda \mathtt{f}.\lambda \mathtt{x}.\ ((\lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{f}^{n}\ \mathtt{x})\ \mathtt{f})^{m}\ \mathtt{x} \\
& \longrightarrow & \lambda \mathtt{f}.\lambda \mathtt{x}.\ (\lambda \mathtt{x}.\ \mathtt{f}^{n}\ \mathtt{x})^{m}\ \mathtt{x} \\
& \longrightarrow & \lambda \mathtt{f}.\lambda \mathtt{x}.\ (\lambda \mathtt{x}.\ \mathtt{f}^{n}\ \mathtt{x})^{m-1}\ (\mathtt{f}^{n}\ \mathtt{x}) \\
& \longrightarrow & \lambda \mathtt{f}.\lambda \mathtt{x}.\ (\lambda \mathtt{x}.\ \mathtt{f}^{n}\ \mathtt{x})^{m-2}\ (\mathtt{f}^{n}\ (\mathtt{f}^{n}\ \mathtt{x})) \\
& \vdots & \\
& \longrightarrow & \lambda \mathtt{f}.\lambda \mathtt{x}.\ (\mathtt{f}^{n}\ )^{m}\ \mathtt{x} \\
& \longrightarrow & \lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{f}^{n \cdot m}\ \mathtt{x} \\
& \equiv & \overline{m \cdot n}
\end{array}
$$

3.3–c **p** $\overline{n} = \overline{m} \quad \equiv \quad \overline{n} =$ **s** $\overline{m}$*:*

Dieser Beweis ist ein wenig umfangreicher. Er wird in zwei Teile aufgespaltet:

1. *Der Hauptbeweis*
2. *Ein Lemma über die Reduktion eines Teilredex'*

Wir überlegen zunächst, wie die Reduktion von $\mathbf{p}\ \overline{n}$ *unabhägig von einem konkreten* n *prinzipiell aussieht.*

Vorüberlegung:

$$
\begin{array}{lcl}
\mathbf{p}\ \overline{n} & \equiv & (\lambda \mathtt{n.}\ (\mathtt{n}\ (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)\ \langle \lambda \mathtt{z.}\overline{0},\ \overline{0}\rangle).\mathbf{2})\ \overline{n} \\
& \longrightarrow & (\overline{n}\ (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)\ \langle \lambda \mathtt{z.}\overline{0},\ \overline{0}\rangle).\mathbf{2} \\
& \equiv & (\ (\lambda \mathtt{f.}\lambda \mathtt{x.}\ \mathtt{f}^n\ \mathtt{x})\ (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)\ \langle \lambda \mathtt{z.}\overline{0},\ \overline{0}\rangle\ ).\mathbf{2} \\
& \longrightarrow & (\ (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)^n\ \langle \lambda \mathtt{z.}\overline{0},\ \overline{0}\rangle\ ).\mathbf{2}
\end{array}
$$

Nun können wir zwei Fälle für diesen Redex unterscheiden:

Erster Fall — $n = 0$**.** *Dann gilt:*

$$
\begin{array}{lcl}
\mathbf{p}\ \overline{0} & \equiv & (\ (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)^0\ \langle \lambda \mathtt{z.}\overline{0},\ \overline{0}\rangle\ ).\mathbf{2} \\
& \equiv & \langle \lambda \mathtt{z}\ \overline{0},\ \overline{0}\rangle.\mathbf{2} \\
& \longrightarrow & \overline{0}
\end{array}
$$

Zweiter Fall — $n = m + 1$**.** *Dann gilt:*

$$
\begin{array}{lcl}
\mathbf{p}\ \overline{m+1} & \equiv & (\ (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)^{m+1}\ \langle \lambda \mathtt{z.}\overline{0},\ \overline{0}\rangle\ ).\mathbf{2} \\
& \equiv & (\ (\ \lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)^m \\
& & \quad (\ (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)\ \ \langle \lambda \mathtt{z.}\ \overline{0},\ \overline{0}\rangle\ )\ ).\mathbf{2} \\
& \longrightarrow & (\ (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)^m \\
& & \quad \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \langle \lambda \mathtt{z.}\ \overline{0},\ \overline{0}\rangle\ \mathbf{in}\ \mathtt{f\ x}\rangle\ ).\mathbf{2} \\
& \longrightarrow & (\ (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)^m\ \langle \mathbf{s},\ (\lambda \mathtt{z.}\ \overline{0})\ \overline{0}\rangle\ ).\mathbf{2} \\
& \longrightarrow & (\ (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)^m\ \langle \mathbf{s},\ \overline{0}\rangle\ ).\mathbf{2} \\
\text{siehe Lemma (*)} & & \\
& \longrightarrow & \langle \mathbf{s},\ \mathbf{s}^m\ \overline{0}\rangle.\mathbf{2} \\
& \longrightarrow & \mathbf{s}^m\ \overline{0} \\
& \longrightarrow & \overline{m}
\end{array}
$$

Induktionsbeweis für Lemma (*) —
$(\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)^n\ \langle \mathbf{s},\ \overline{0}\rangle \equiv \langle \mathbf{s},\ \mathbf{s}^n\ \overline{0}\rangle$*:*

Anfang — $n = 0$*:*

$(\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)^0\ \langle \mathbf{s},\ \overline{0}\rangle \equiv \langle \mathbf{s},\ \overline{0}\rangle \equiv \langle \mathbf{s},\ \mathbf{s}^0\ \overline{0}\rangle$ $\surd$

Schritt: *es gelte* $(\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)^n\ \langle \mathbf{s},\ \overline{0}\rangle \equiv \langle \mathbf{s},\ \mathbf{s}^n\ \overline{0}\rangle$.

Dann folgt:

$$
\begin{array}{lcl}
& & (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)^{n+1}\ \ \langle \mathbf{s},\overline{0}\rangle \\
& \longrightarrow & (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle) \\
& & \quad (\ (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)^n\ \ \langle \mathbf{s},\ \overline{0}\rangle\ ) \\
\text{wegen Annahme} & & \\
& \equiv & (\lambda \mathtt{z.}\ \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \mathtt{z}\ \mathbf{in}\ \mathtt{f\ x}\rangle)\ \ \langle \mathbf{s},\ \mathbf{s}^n\ \overline{0}\rangle \\
& \longrightarrow & \langle \mathbf{s},\ \mathbf{let}\ \langle \mathtt{f,x}\rangle = \langle \mathbf{s},\ \mathbf{s}^n\ \overline{0}\rangle\ \mathbf{in}\ \mathtt{f\ x}\rangle \\
& \longrightarrow & \langle \mathbf{s},\ \mathbf{s}\ (\mathbf{s}^n\ \overline{0})\rangle \\
& \equiv & \langle \mathbf{s},\ \mathbf{s}^{n+1}\ \overline{0}\rangle
\end{array}
$$

Es gibt einen kürzeren Beweis ... wenn ich Zeit finde, schreibe ich ihn auf

Lösung 3.4 *Ziel dieser Aufgabe ist es, in bezug auf das oben formulierte Gesamtziel dieses Übungsblattes in die Vollen zu gehen. Als Seiteneffekt werden dabei nochmals die Inhalte der Theoretischen Informatik ein Stück weit aufgewärmt.*

3.4–a **subtract***:*

*Definitionsgemäß unterscheiden wir zwei Fälle für den zweiten Eingabeparameter $\overline{y}$: $y = 0$ oder $y > 0$. Im ersten Fall geben wir den ersten Eingabeparameter ($\overline{x}$) zurück, im zweiten gehen wir davon aus, daß wir für den Vorgänger von $\overline{y}$ das Ergebnis kennen und wenden auf dieses die Vorgängerfunktion an. Dies klingt nun verdächtig nach einem Fall für den einfachen Rekursionsoperator "***PRs***[$base$, h]":*

*Für "$base$" muß somit der erste Parameter "$\overline{x}$" herhalten und für "h" demzufolge "**p**". Das Ganze sieht dann so aus:*

$$\textbf{subtract} \equiv \lambda \mathtt{x}.\lambda \mathtt{y}.\textbf{PRs}[\mathtt{x},\textbf{p}]\ \mathtt{y}$$

Wem's Spaß macht, der kann ja mal den "nackten" λ–Term dazu hinschreiben.

Subtraktion von n ist n-fache Anwendung der Vorgängerfunktion p*. Da das Church Numeral $\overline{n}$ als die n-fache Anwendung einer Funktion auf ein Argument codiert ist, kann der Term direkt angegeben werden als* sub $\equiv$ *$\lambda m.\lambda n.$ n* p *m (Achtung: implizite Linksklammerung)*

Es ist

$$
\begin{aligned}
&\text{sub}\ \overline{m}\ \overline{n}\ \equiv (\lambda m.\lambda n.\ n\ \text{p}\ m)\ \overline{m}\ \overline{n}\\
&\longrightarrow \overline{n}\ \text{p}\ \overline{m}\\
&\equiv (\lambda f.\lambda x.f^n\ x)\ \text{p}\ \overline{m}\\
&\longrightarrow \text{p}^n\ \overline{m}\\
&\xrightarrow{*} \overline{m-n}, \quad \textit{weil}\ \text{p}\ \overline{m} = \overline{m-1}\ \textit{gilt.}
\end{aligned}
$$

3.4–b **less_or_equal***:*

Als gestandener Informatiker sollte man sich klar machen können, daß $m \leq n$ äquivalent zu $m - n \leq 0$ (bzw. $m \dot{-} n = 0$ für nicht–negative Zahlen) ist. Damit kann man die Definition für **less_or_equal** *auch so hinschreiben:*

$$\textbf{less\_or\_equal}\ \overline{m}\ \overline{n} \equiv \begin{cases} \boldsymbol{T}, & \textit{falls } m \dot{-} n = 0 \\ \boldsymbol{F}, & \textit{sonst} \end{cases}$$

*An dieser Stelle sollte es beim geneigten Leser bereits klingeln: das hierfür notwendige Werkzeug haben wir nämlich längst zur Hand. Es ist dies die "***zero***"–Funktion in Kombination mit dem eben definierten "***subtract***". Damit gelangen wir also zu folgendem Resultat:*

$$\textbf{less\_or\_equal} \equiv \lambda \mathtt{m}.\lambda \mathtt{n}.\textbf{zero}\ (\textbf{subtract}\ \mathtt{m}\ \mathtt{n})$$

Hier darf wiederum den kompletten λ–Term aufschreiben, wem's Freude bereitet...

Es ist $m < n$ falls $m - n < 0$ bzw. falls $(m + 1) - n \leq 0$. Da die Subtraktion niemals Werte unter Null annehmen kann, reicht es zu testen, ob $(m + 1) - n = 0$ ist, wofür wir die Funktion zero *verwenden.*
less $\equiv$ *$\lambda m.\lambda n.$* zero (sub (s *m*) *n*)

3.4–c **max***:*

Hier suggeriert die Definition die Anwendung eines Conditionals gemäß folgender Formulierung:

"**Wenn** $m \leq n$ *gilt*, **dann** *ist* **max** $\overline{m}\ \overline{n}$ *gleich* $\overline{n}$, **ansonsten** *ist* **max** $\overline{m}\ \overline{n}$ *gleich* $\overline{m}$."

Damit gelangen wir zu folgendem Term:

$$
\begin{aligned}
\textbf{max} \quad &\equiv \quad \lambda \mathtt{m}.\lambda \mathtt{n}.\textbf{if less\_or\_equal}\ \mathtt{m}\ \mathtt{n}\ \textbf{then}\ \mathtt{n}\ \textbf{else}\ \mathtt{m}\\
&\equiv \quad \lambda \mathtt{m}.\lambda \mathtt{n}.\textbf{cond}(\textbf{less\_or\_equal}\ \mathtt{m}\ \mathtt{n};\mathtt{n};\mathtt{m})\\
&\equiv \quad \lambda \mathtt{m}.\lambda \mathtt{n}.(\textbf{less\_or\_equal}\ \mathtt{m}\ \mathtt{n})\ \mathtt{n}\ \mathtt{m}
\end{aligned}
$$

Insgesamt sollte man hierbei erkannt haben, wie wichtig die definitorische Erweiterung ist, wenn man vernünftig mit dem λ–Kalkül arbeiten will. Wer das (immer noch) nicht glaubt, der kann ja mal versuchen, dieselbe Herleitung mit den »nackten« λ–Termen durchzuexerzieren...