Karlsruher Institut für Technologie

Institut für Programmstrukturen und Datenorganisation
Prof. Dr.-Ing. Gregor Snelting   Prof. Dr. Ralf Reussner
gregor.snelting@kit.edu   reussner@kit.edu

Programmierparadigmen – WS 2021/22

https://pp.ipd.kit.edu/lehre/WS202122/paradigmen/uebung

Blatt 5: λ-Kalkül, Church-Kodierungen

Abgabe:	26.11.2021, 14:00
Besprechung:	29.11. – 30.11.2021

Reichen Sie Ihre Abgabe bis zum 26.11.2021 um 14:00 in unserer Praktomat-Instanz unter https://praktomat.cs.kit.edu/pp_2021_WS ein.

Hinweis: Im Rahmen von PSE wurde in einem vorherigen Semester ein Lerntool entwickelt, welches λ-Ausdrücke reduzieren und strukturell darstellen kann. Dieses kann Ihnen beim Lösen dieser Aufgaben behilflich sein.[1]

1 Klammerung im λ-Kalkül [größtenteils aus "To Mock a Mockingbird" von Raymond Smullyan]

1. Fügen Sie in die folgenden Ausdrücke alle impliziten Klammern ein!
   (a) $c_0\ c_1\ (c_2\ c_3\ c_4)\ c_5$
   (b) $(c_0\ c_1\ c_2)\ (c_3\ c_4\ c_5)$
   (c) $c_0\ c_1\ (c_2\ c_3\ c_4)\ (c_5\ c_6)$
   (d) $c_0\ c_1\ (c_2\ c_3\ c_4)\ c_5\ c_6$
   (e) $c_0\ (c_1\ (c_2\ c_3\ c_4))\ c_5\ c_6$
   (f) $(\lambda y.\ c_0\ c_1\ c_2)\ (c_3\ c_4\ c_5)$
   (g) $(\lambda y.\ c_0\ (\lambda z.\ c_1\ c_2))\ (c_3\ c_4\ c_5)$
2. Welcher dieser beiden λ-Terme repräsentiert den gleichen λ-Term wie $\lambda y.\ y\ c_0$?
   (a) $(\lambda y.\ y)\ c_0$
   (b) $\lambda y.\ (y\ c_0)$

   (a) Führen Sie in folgenden Termen Substitution durch:
      i. $x = \lambda y.\ y$ in den Term $(x)\ c_0$
      ii. $x = (\lambda y.\ y)$ in den Term $x\ \ c_0$
   (b) Gilt folgende Aussage im λ-Kalkül?

      "Für beliebiges t repräsentieren t und (t) den gleichen λ-Term"
   (c) Führen Sie in folgendem Term Substitution durch:[2]
      iii. $x = \lambda y.\ y$ in den Term $x\ \ c_0$

[1] Siehe https://pp.ipd.kit.edu/lehre/WS201718/paradigmen/lambda-ide/Wavelength.html. Das Tool ist zudem auf der Webseite der Übung verlinkt

[2] Betrachten Sie noch einmal die vorherigen Teilaufgaben. Haben Sie konsistent geantwortet?

3. Angenommen, $x = c_0\ c_1$. Welche der folgenden Aussagen gelten?
   (a) $c_0\ c_1\ c_2 = x\ c_2$
   (b) $c_2\ c_0\ c_1 = c_2\ x$
   (c) $c_2\ (c_3\ c_4)\ c_0\ c_1 = c_2\ (c_3\ c_4)\ x$
   (d) $c_2\ (c_0\ c_1\ c_3)\ c_4 = c_2\ (x\ c_3)\ c_4$
4. Unterstreichen Sie alle *linken Seiten* der Redexe (also die "sofort anwendbaren Funktionen") in folgendem Term:

   $$(\lambda a.a)\ (\lambda b.b)\ ((\lambda c.c)\ ((\lambda d.d)\ (\lambda e.e)\ (\lambda f.f)))\ (\lambda g.g)\ ((\lambda h.h)\ (\lambda i.i)).$$

   Führen Sie dann, jeweils ausgehend von obigem Term, je einen Reduktionsschritt pro Redex durch.

   Statt zu unterstreichen können Sie auch einfach den Variablennamen der zugehörigen Lambda-Abstraktion angeben.

2 Redexe, Auswertungsstrategie

Markieren Sie in den folgenden λ-Termen alle vorkommenden Redexe. Welcher Redex wird unter den Auswertungsstrategien *Normalreihenfolge*, *Call-By-Name* bzw. *Call-By-Value* jeweils als nächstes reduziert?

$t_1 = (\lambda\texttt{t. }\lambda\texttt{f. f})\ ((\lambda\texttt{y. }(\lambda\texttt{x. x x})\ (\lambda\texttt{x. x x}))\ ((\lambda\texttt{x. x})\ (\lambda\texttt{x. x})))\ (\lambda\texttt{t. }\lambda\texttt{f. f})$

$t_2 = \lambda\texttt{y. }(\lambda\texttt{z. }(\lambda\texttt{x. x})\ (\lambda\texttt{x. x})\ \texttt{z})\ \texttt{y}$

Sie können diese Aufgabe eingescannt, abfotografiert oder z.B. als ASCII-Art abgeben:

```
(\x. x) ((\y. y) z)
         ------- ~  CBV
------- ~~~~~~~~~~~ NRF, CBN
```

In den meisten PDF-Readern sollten Sie die Terme auch 1:1 aus den Aufgaben herauskopieren können.

3 Church-Zahlen in Haskell [Klausuraufgabe vom WS 2010/11] [10 Punkte]

Reichen Sie Ihre Lösung als Modul `ChurchNumbers` ein.

Bekanntlich kann man die natürlichen Zahlen im λ-Kalkül codieren als:

$$c_n \equiv \lambda\texttt{s. }\lambda\texttt{z. }\underbrace{\texttt{s (... s (s z))}}_{\text{n mal}}$$

Diese Church-Zahlen haben den (polymorphen) Typ

```
type Church t = (t -> t) -> t -> t.
```

Geben Sie zwei Haskellfunktionen

```
int2church :: Integer -> Church t
church2int :: Church Integer -> Integer
```

an, die gewöhnliche natürliche Zahlen in Church-Zahlen verwandeln und umgekehrt.
Hinweis: `church2int` lässt sich ohne Rekursion definieren.

4 Church-Paare [weitgehend aus alten Klausuraufgaben]

Aus der Vorlesung kennen Sie die Modellierung von Church-Zahlen und Church-Booleans. Auf ähnliche Weise kann man auch Paare von Elementen im λ-Kalkül darstellen. Beispielsweise lässt sich das Paar $(3, 5)$ (unter Verwendung von Church-Zahlen c_n) darstellen als

$$\lambda \mathtt{f}.\ \mathtt{f}\ c_3\ c_5$$

Der Paar-Konstruktor *pair* ist damit:

$$\mathit{pair} = \lambda \mathtt{a}.\ \lambda \mathtt{b}.\ \lambda \mathtt{f}.\ \mathtt{f}\ \mathtt{a}\ \mathtt{b}$$

Wichtige Funktionen auf Paaren sind die Destruktoren *fst* und *snd*, die jeweils das erste bzw. zweite Element aus dem Paar extrahieren. Die Definition von *fst* ist:

$$\mathit{fst} = \lambda \mathtt{p}.\ \mathtt{p}\ (\lambda \mathtt{a}.\ \lambda \mathtt{b}.\ \mathtt{a})$$

1. Definieren Sie *snd*. [1 Punkt]
2. Zeigen Sie mit Beta-Reduktion unter Verwendung von Call-By-Name: [4 Punkte]
$$\mathit{fst}\ (\mathit{pair}\ a\ b) \Rightarrow^* a$$
3. Geben Sie einen Lambdaterm *next* an, der zur Church-Darstellung von (n, m) die Church-Darstellung von $(m, m + 1)$ berechnet. [3 Punkte]
4. Zeigen Sie, dass für beliebige Terme n, m gilt: $\mathit{next}\ (\mathit{pair}\ n\ m) \Rightarrow^* \mathit{pair}\ m\ (\mathit{succ}\ m)$ [2 Punkte]
5. Verwenden Sie *next*, um eine Vorgängerfunktion *pred* für Church-Zahlen anzugeben. [5 Punkte]
Hinweis: n-malige Anwendung von *next* auf $\mathit{pair}\ c_0\ c_0$ ergibt $\mathit{pair}\ c_{n-1}\ c_n$
6. Berechnen Sie den Vorgänger von c_2, indem Sie die zugehörige Beta-Reduktion angeben. [5 Punkte]
7. Definieren Sie eine Subtraktionsfunktion *sub*, die zwei Church-Zahlen voneinander abzieht. Was ist die Normalform von $\mathit{sub}\ c_2\ c_1$? [Nicht Teil der Klausur]

5 B-Seite: Omega and beyond...

Aus der Vorlesung kennen Sie bereits $\omega = (\lambda\texttt{x. x x})\ (\lambda\texttt{x. x x})$, einen divergierenden Term. In dieser Aufgabe befassen wir uns näher mit solchen Termen.

1. Führen Sie volle β-Reduktion für folgende Terme aus:
   (a) $(\lambda\texttt{x. x x})\ (\lambda\texttt{y. m y n})$
   (b) $(\lambda\texttt{x. x x})\ (\lambda\texttt{y. f y})$
   (c) $(\lambda\texttt{x. x x})\ (\lambda\texttt{y. (f y) y})$

2. (a) Führen Sie 4 β-Reduktionsschritte für folgenden Term aus:
   $$\omega' = (\lambda\texttt{x. x x})\ (\lambda\texttt{y. m (y y) n})$$
   Hinweis: Sparen Sie sich Schreibarbeit, indem Sie sich Abkürzungen für wiederholte Subterme definieren.

   (b) Es sei
   $$\mathit{count} = \lambda\texttt{y}.\lambda\texttt{n}.\ \mathit{isZero}\ \texttt{n}\ c_1\ ((\texttt{y y})\ (\mathit{sub}\ \texttt{n}\ c_1)).$$
   Reduzieren Sie folgenden Term vollständig unter Verwendung der Call-by-Name-Reihenfolge:
   $$(\lambda\texttt{x. x x})\ \mathit{count}\ c_2$$
   Hinweis: Sparen Sie sich Schreibarbeit, wo Sie können:
   - Nehmen Sie an, dass *isZero* und *sub* so funktionieren, wie man es erwarten würde, ohne die Definitionen auszufalten.
   - Wenn Sie einen bekannten Subterm nochmal reduzieren müssen, können Sie gleich das Endresultat einsetzen.

   (c) Wie viele Reduktionsschritte kann man auf ω' anwenden? Machen Sie sich die Rolle des Subterms $\texttt{y y}$ in diesen Reduktionen klar.

3. (a) Geben Sie einen Term W an, welcher unter β-Rekursion nach und nach die Form $\texttt{f (f (f (f (...))))}$ annimmt.
   (b) Beschreiben Sie, welche Form der Term $X = \lambda\texttt{f}.\ (\lambda\texttt{x. x x})\ (\lambda\texttt{x. f (x x)})$ unter β-Reduktion annimmt.
   (c) Führen Sie einen β-Reduktionsschritt von X aus. Erkennen Sie den resultierenden Term wieder?

4. Geben Sie unter Benutzung von X einen Term an, welcher zu ω reduziert