

Übungen zur Vorlesung
 „Grundlagen der Programm- und Systementwicklung“

Aufgabe 8.1 (H) Ungetypter λ -Kalkül

In der Vorlesung haben Sie ein Verfahren kennengelernt, mit dem Funktionen verwendet werden können, ohne ihnen einen Namen zu geben. Verzichtet man auf die Typangaben, so ergibt sich die Definition des *ungetypten λ -Kalküls*: Über einer abzählbaren Menge von Variablen Var sind die Ausdrücke $Expr$ des λ -Kalküls über

Expr ::=	v	$(v \in Var)$	Variablen
	$ (e1\ e2)$	$(e1, e2 \in Expr)$	Applikation
	$ (\lambda v.e)$	$(v \in Var, e \in Expr)$	Abstraktion

definiert. Zur Vereinfachung wird vereinbart, daß die Applikation linksassoziativ, d.h. $(x\ y\ z) = ((x\ y)\ z)$, und der Wirkungsbereich von λ so weit rechts wie möglich ist, d.h. $(\lambda x.x\ x)$ steht für $(\lambda x.(x\ x))$ und nicht für $((\lambda x.x)\ x)$. Außerdem wird $\lambda x.\lambda y.e$ kurz als $\lambda xy.e$ geschrieben. Zur Unterscheidung von (durch λ) gebundenen und freien Variablen werden die Funktionen *free*, *bound*: $Expr \rightarrow \wp(Var)$ wie folgt definiert:

$free(v) = \{v\}$ für $v \in Var$, $free(e1\ e2) = free(e1) \cup free(e2)$, $free(\lambda x.e) = free(e) \setminus \{x\}$,
 $bound(v) = \emptyset$ für $v \in Var$, $bound(e1\ e2) = bound(e1) \cup bound(e2)$, $bound(\lambda x.e) = bound(e) \cup \{x\}$.

Für $e, f, g \in Expr$, $x \neq v \in Var$ bedeutet $e[v/f]$, daß v in e durch f ersetzt wird: $v[v/f] = f$, $x[v/f] = x$,
 $(e\ g)[v/f] = (e[v/f]\ g[v/f])$, $(\lambda v.e)[v/f] = \lambda v.e$, $(\lambda x.e)[v/f] = \lambda x.e[v/f]$ für $x \notin free(f)$,
 $(\lambda x.e)[v/f] = \lambda y.(e[x/y][v/f])$ für $x \in free(f)$, $y \notin free(f) \cup free(e)$.

Im λ -Kalkül gibt es u.a. zwei Reduktionsarten, die α - und die β -Reduktion. Die Reduktionsrelation $\rightarrow_\beta \subseteq Expr \times Expr$ ist durch $(\lambda v.e)\ f \rightarrow_\beta e[v/f]$ definiert und entspricht einer Ersetzung formaler Parameter (v) durch aktuelle (f). Zum Beispiel ist $(\lambda xy.\ x+y)\ 1\ 2 \rightarrow_\beta 1+2$. Die Reduktionsrelation $\rightarrow_\alpha \subseteq Expr \times Expr$ ist für $y \in free(e)$ durch $\lambda x.e \rightarrow_\alpha \lambda y.e[x/y]$ definiert und entspricht damit einer Variablenumbenennung. Zum Beispiel ist $(\lambda x.f\ x) \rightarrow_\alpha \lambda y.f\ y$. Da bisher beide Reduktionen nur auf das „äußerste“ λ angewendet werden können, wird \rightarrow_β (und analog \rightarrow_α) wie folgt erweitert: $\forall e, f, g: e \rightarrow_\beta f \Rightarrow e\ g \rightarrow_\beta f\ g \wedge g\ e \rightarrow_\beta g\ f \wedge \lambda x.e \rightarrow_\beta \lambda x.f$.

1. Machen Sie sich den Zusammenhang zwischen dieser Notation und der in der Vorlesung verwendeten klar!
2. Bestimmen Sie zu allen Teilausdrücken von $((\lambda x.(x\ (\lambda y.(x\ ((\lambda y.(y\ x))\ y))))\ y))\ x$ die freien und die gebundenen Variablen!
3. Welche Möglichkeiten gibt es, $(\lambda x.I)((\lambda x.(x\ x))\ (\lambda x.(x\ x)))$ zu reduzieren?
4. Zeigen Sie, daß mit den sog. Kombinatoren $True := \lambda xy.x$, $False := \lambda xy.y$ und $Cond := \lambda bxy.bxy$ if-then-else-Konstrukte simulieren werden können!
5. Gibt es endliche Terme, die unendlich oft β -reduziert werden können? Wenn ja, nennen Sie Beispiele! Gibt es endliche Terme im getypten λ -Kalkül, die unendlich oft β -reduziert werden können? Beispiele!

Aufgabe 8.2 (H) Interpreter in ML für den ungetypten λ -Kalkül

Für die Implementierung eines Interpreters sollen Variablen als Strings und λ -Ausdrücke als

Konstruktorterme dargestellt werden: `type variable = string` und
`datatype lambdaausdruck = var of variable`
| `lambda of variable*lambdaausdruck`
| `appl of lambdaausdruck*lambdaausdruck`

1. Definieren Sie sich einen Datentypen `Menge` `alpha`, der Mengen mit den üblichen Operationen implementiert! Implementieren Sie eine Funktion `listeNachMenge`, die eine Liste in eine Menge konvertiert!
2. Definieren Sie Funktionen `free:lambdaausdruck→Menge(variable)`, `bound:lambdaausdruck→Menge(variable)`, die oben erklärt wurden. Bei Namenskonflikten muß eine „frische Variable“ verwendet werden; das erledigt eine Funktion `neuevariable:string→Menge(string) →string`, die eine Variable und eine Menge bereits verwendeter Variablen erhält und eine neue Variable durch Anhängen einer Zahl liefert, z.B. ergibt `neuevariable „x“ (listeNachMenge [„x1“ „y“ „x2“])` die neue Variable „x3“.

`subst:lambdaausdruck→variable→lambdaausdruck→lambdaausdruck` ersetzt die freien Vorkommen einer Variablen in einem λ -Ausdruck durch einen anderen λ -Ausdruck. Schließlich führt `beta:lambdaausdruck→lambdaausdruck` eine β -Reduktion in einem Ausdruck durch, wobei der „linksäußerste“ reduzierbare Term gewählt wird, wenn mehrere Möglichkeiten bestehen („linksäußerst“ bezieht sich auf die Baumdarstellung von Termen“). Die Funktion `betaStar:lambdaausdruck→lambdaausdruck` schließlich reduziert einen Term so lange, bis es nicht mehr geht („bis er in Normalform ist“, falls es eine solche gibt).

Aufgabe 8.3 (H) Church-Numerale

Man kann die natürlichen Zahlen durch Funktionale definieren. Dabei entspricht eine natürliche Zahl der n -fachen Anwendung einer Funktion auf ein beliebiges, aber festes Element: `Null := $\lambda f.\lambda x.x$` , `Eins := $\lambda f.\lambda x.f x$` , `Zwei := $\lambda f.\lambda x.f f x$` usw.

1. Definieren Sie mit Hilfe dieser sog. *Church-Numerale* Addition, Multiplikation und Exponentiation! Läßt sich die Subtraktion (oder Dekrement) ähnlich einfach definieren? Begründen Sie Ihre Antwort!
2. Testen Sie Ihre Definitionen mit dem oben erstellten Interpreter!

Aufgabe 8.4 (H) Fixpunktkombinator

Der Fixpunktkombinator `Y` (entspricht dem τ der Vorlesung) ist wie folgt definiert:

`Y:= $\lambda f. (\lambda x. f (x x)) (\lambda x. f (x x))$` . Zeigen Sie, daß `Y F=F (Y F)` für alle `F` gilt! Versuchen Sie, `Y` im getypten λ -Kalkül (d.h. in der Notation der Vorlesung) zu formalisieren. Warum scheitern Sie?

Aufgabe 8.5 (H) Funktionale Spezifikation

Spezifizieren Sie möglichst abstrakt Funktionen, die (1) zu einem Graphen und einem Knoten die Menge aller von diesem Knoten erreichbaren Knoten, (2) die transitive Hülle eines Graphen und (3) alle geschlossenen Kantenzüge der Länge n berechnen!

Wir wünschen Ihnen fröhliche Weihnachten und einen guten Rutsch ins neue Jahr!