Lambda calculus

Petr Štěpánek
Department of Theoretical Computer Science
and Mathematical Logic
Charles University, Prague

Based on material provided by H. P. Barendregt

Alonzo Church 1932

Haskell B Curry 1935-1981

Dana S. Scott 1969

The functional computational model

A functional program - an expression ***E***

Rule $P \rightarrow P'$

Reduction $E[P] \rightarrow E[P']$

Output ***E****

Example

$$(8-5)*(6+2*3) \rightarrow 3*(6+2*3)$$
$$\rightarrow 3*(6+6)$$
$$\rightarrow 3*12$$
$$\rightarrow 36$$

$$(8-5)*(6+2*3) \rightarrow (8-5)*(6+6)$$
$$\rightarrow (8-5)*12$$
$$\rightarrow 3*12$$
$$\rightarrow 36$$

Lambda Calculus

Part I

Untyped calculus

Conversion

Application

The data (expression) ***F*** considered as an algorithm are applied to the data ***A*** considered as input. Notation

$$(FA)$$

Abstraction

Let $\boldsymbol{M[x]}$ be an expression possibly depending on a variable $\boldsymbol{x}$, then

$$\lambda x.M[x]$$

denotes the map

$$x \mapsto M[x]$$

Example

$$(\lambda x.x + 2)3 = 3 + 2 = 5$$

(β) $$(\lambda x.M[x])N = M[x := N]$$

The left-hand side of (β) is called *redex* and the right-hand side is called *contractum*.

Free and bound variables

Abstraction is said to bind the free variables in *M*. Substitution [x:=N] is only performed in the free occurences of x:

$$(\lambda y.xy)(\lambda x.x)[x:=N]=(\lambda y.Ny)(\lambda x.x)$$

Functions of several arguments

can be obtained by iteration of application.

If we have

$$f(x,y)$$

Put

$$F_x=\lambda y.f(x,y)$$
$$F=\lambda x.F_x$$

then

$$(Fx)y=F_xy=f(x,y) \qquad (1)$$

The equation (1) shows that it is convenient to associate parentheses to the left for iterated application:

$$FM_1M_2\ldots M_n \quad \text{denotes} \quad (\ldots((FM_1)M_2)\ldots M_n)$$

then (1) becomes

$$Fxy=f(x,y)$$

On the other hand it is convenient to use association parentheses to the right for the iterated abstraction

$$\lambda x_1x_2\ldots x_n.f(x_1,x_2\ldots x_n) \quad \text{denotes}$$
$$\lambda x_1.(\lambda x_2.(\ldots(\lambda x_n.f(x_1,x_2\ldots x_n))\ldots))$$

Then we have for F defined above

$$F=\lambda xy.f(x,y)$$

and (1) becomes

$$(\lambda xy.f(x,y))xy=f(x,y)$$

Formal description of lambda calculus

The set of lambda terms Λ is built up from infinite sets of constants and variables using application and abstraction.

$$C = \{c, c', c'', \ldots\} \qquad V = \{v, v', v'', \ldots\}$$

$$c \in C \Rightarrow c \in \Lambda \qquad x \in V \Rightarrow x \in \Lambda$$

$$M, N \in \Lambda \Rightarrow (MN) \in \Lambda$$

$$M \in \Lambda, x \in V \Rightarrow (\lambda x.M) \in \Lambda$$

Description by abstract syntax

$$\Lambda = C \mid V \mid \Lambda\Lambda \mid \lambda V \Lambda$$

Example

$$v \quad (vc) \quad (\lambda v(vc)) \quad (v'(\lambda v(vc))) \quad ((\lambda v(vc))v')$$

are Λ terms.

The set FV(M) of free variables of M

$$FV(x) = \{x\} \qquad FV(MN) = FV(M) \cup FV(N)$$

$$FV(\lambda x.M) = FV(M) - \{x\}$$

M is a *closed λ-term* if $FV(M) = 0$

The set of closed λ-terms is denoted Λ_0.

Lambda calculus as a theory of equation

The principal axiom scheme

$$(\beta) \qquad (\lambda x.M)N = M[x := N]$$

for all $M, N \in \Lambda$.

The logical axioms and rules

$$M = M$$

$$M = N \Rightarrow N = M$$

$$M = N, N = L \Rightarrow M = L$$

$$M = M' \Rightarrow MZ = M'Z$$

$$M = M' \Rightarrow ZM = ZM'$$

$(rule-\xi)$ $\quad M = M' \Rightarrow (\lambda x.M) = (\lambda x.M')$

If $M = N$ is provable from the axioms and rules we write

$$\lambda \mid\!- M = N$$

or just say that M and N are β-convertible.

$M \equiv N$ denotes that M and N are the same term or can be obtained from each other by renaming bound variables.

Examples

$(\lambda x.y)z \equiv (\lambda x.y)z \qquad (\lambda x.x)z \equiv (\lambda y.y)z \qquad (\lambda x.x)z \not\equiv (\lambda x.y)z$

An alternative

$(\alpha) \quad (\lambda x.M) = (\lambda y.M)[x := y]$ $\qquad y$ does not occur in M.

Name-free notation

$\lambda x.(\lambda y.xy)$ is denoted by $\lambda(\lambda 21)$

Development of the theory

standard combinators

$I \equiv \lambda x.x \qquad K \equiv \lambda xy.x$

$K_* \equiv \lambda xy.y \qquad S \equiv \lambda xyz.xz(yz)$

We have

$IM = M \qquad KMN = M$

$K_*MN = N \qquad SMNL = ML(NL)$

Fixed Point Theorem

(i) For every $F \in \Lambda$ there is an $X \in \Lambda$ such that

$$\lambda \mid - FX = X$$

$$(\forall F \, \exists X \; FX = X)$$

(ii) There is a fixed point combinator

$$\mathbf{Y} \equiv \lambda f.(\lambda x. f(xx))(\lambda x. f(xx))$$

such that $\forall F \; F(\mathbf{Y}F) = \mathbf{Y}F$

Proof. (i) Define $W \equiv \lambda x.F(xx)$ and $X = WW$

Then

$$X \equiv WW \equiv (\lambda x.F(xx))W = F(WW) \equiv FX$$

(ii) By the proof of (i)

A term C[f,x] possibly containing the displayed variables is called a context.

Context lemma.

Given a context C[f,x], we have

$$\exists F \, \forall X \; FX = C[F, X]$$

Where C[F,X] is the result of the substitution

$$C[f,x][f{:=}F][x{:=}X]$$

Proof.

$$\begin{aligned}
\forall X \; FX = C[F, X] &\Leftarrow Fx = C[F, x] \\
&\Leftarrow F = \lambda x.C[F, x] \\
&\Leftarrow F = (\lambda fx.C[f, x])F \\
&\Leftarrow F = \mathbf{Y}(\lambda fx.C[f, x]).
\end{aligned}$$

Definition

Let N be the set of natural numbers and $n \in N$.

We define

$$F^0(M) \equiv M \qquad F^{n+1}(M) \equiv F(F^n(M))$$

Definition The Church numerals

$$c_0, c_1, c_2, \ldots, c_n, \ldots$$

are defined by

$$c_n \equiv \lambda fx. f^n(x)$$

Lemma (Rosser)

If we define

$$A_+ \equiv \lambda xypq.xp(ypq)$$
$$A_* \equiv \lambda xyz.x(yz)$$
$$A_{\mathbf{exp}} \equiv \lambda xy.yx$$

We have for every $n, m \in N$

$$A_+ c_n c_m = c_{n+m}$$
$$A_* c_n c_m = c_{n*m}$$
$$A_{\mathbf{exp}} c_n c_m = c_{(n^m)} \quad \text{except for } m = 0.$$

Definition

Put $true \equiv K \qquad false \equiv K_*$

If a term ***B*** is either ***true*** or ***false***, we call it Boolean and for terms ***P, Q*** define

if B then P else Q

as ***BPQ***

indeed

$$truePQ \equiv KPQ = P \qquad falsePQ \equiv K_*PQ = Q$$

Definition (ordered pairs)

For terms M,N write

$$[M, N] = \lambda z.zMN$$

and call it the ordered pair of M and N.

Indeed

$$[M, N]true = M \qquad [M, N]false = N$$

Definition (numerals)

For natural number n define

$$\ulcorner 0 \urcorner \equiv I, \qquad \ulcorner n+1 \urcorner \equiv [false, \ulcorner n \urcorner]$$

Lemma (successor, predecessor, test for zero)

There are combinators $S^{+}, P^{-}, zero$

such that for all $n \in N$

one has

$$S^{+}\ulcorner n \urcorner = \ulcorner n+1 \urcorner \qquad P^{-}\ulcorner n+1 \urcorner = \ulcorner n \urcorner$$
$$zero \ulcorner 0 \urcorner = true \qquad zero \ulcorner n+1 \urcorner = false$$

Proof. Put

$$S^{+} \equiv \lambda x.[false, x]$$
$$P^{-} \equiv \lambda x.x\ false$$
$$zero \equiv \lambda x.x true$$

Definition

A function $f : N^{p} \rightarrow N$ of p arguments is called λ-*definable* if there is a combinator F such that

$$F \ulcorner n_1 \urcorner \ldots \ulcorner n_p \urcorner = \ulcorner f(n_1, \ldots n_p) \urcorner$$

In this case, we say that f is λ-*defined by* F.

Theorem. All recursive functions are λ-definable.

Idea. It can be shown that all basic functions of the class of all recursive functions are λ-definable, that the class of all λ-definable functions is closed under composition, primitive recursion and minimalization.

a) Basic functions

$$U_n^i(x_1,\dots,x_n) = x_i \qquad 1 \le i \le n$$
$$S^+(n) = n+1 \qquad Z(n) = 0$$

Put

$$U_n^i \equiv \lambda x_1,\dots x_n.x_i \qquad S^+ \equiv \lambda x.[false, x] \qquad Z \equiv \lambda x.\ulcorner 0 \urcorner$$

b) Operations.

b1) Composition. Let $g, h_1, \dots, h_m$ be functions λ-defined by $G, H_1, \dots, H_m$ respectively. Then the function

$$f(\vec{n}) = g(h_1(\vec{n}),\dots,h_m(\vec{n}))$$

is λ-defined by

$$F \equiv \lambda\vec{x}.G(H_1\vec{x})\dots(H_m\vec{x}).$$

b2) primitive recursion

Let f be defined by

$$f(0,\vec{n}) = g(\vec{n}), \qquad f(k+1,\vec{n}) = h(f(k,\vec{n}),k,\vec{n})$$

Let g, h be λ-defined by G, H respectively. An intuitive algorithm to compute $f(k,\vec{n})$ consists of the following steps:

- test whether $k = 0$
- if yes, then return $g(\vec{n})$
- if no, then compute $h(f(k-1,\vec{n}),k-1,\vec{n})$

Thus we need a combinator F such that

$$Fx\vec{y} = \mathit{if}\ (\mathbf{Zero}\ x)\ \mathit{then}\ G\vec{y}\ \mathit{else}\ H(F(P^{-}x)\vec{y})(P^{-}x)\vec{y}$$
$$\equiv D(F,x,\vec{y})$$

Now such an combinator F can be found by the Context lemma applied to $D(F,x,\vec{y})$.

b3) (regular) minimalization

$f(\vec{n}) = \mu m[g(\vec{n},m) = 0]$ with $g, \forall\vec{n}\,\exists m\ g(\vec{n},m) = 0$

Suppose that g is λ-defined by G, by the Context lemma there is a term H such that

$$H\vec{x}y = \mathit{if}\,(\mathit{zero}\,(G\vec{x}y))\ \mathit{then}\ y\ \mathit{else}\ H\vec{x}(S^{+}y)$$

Set $F = \lambda\vec{x}.H\vec{x}\lceil 0 \rceil$. Then F λ-defines f, indeed

$$\begin{aligned}
F\lceil\vec{n}\rceil &= H\lceil\vec{n}\rceil\lceil 0\rceil \\
&= \begin{cases} \lceil 0\rceil & \text{if} \quad G\lceil\vec{n}\rceil\lceil 0\rceil = \lceil 0\rceil \\ H\lceil\vec{n}\rceil\lceil 1\rceil & \text{else} \end{cases} \\
&= \begin{cases} \lceil 1\rceil & \text{if} \quad G\lceil\vec{n}\rceil\lceil 1\rceil = \lceil 0\rceil \\ H\lceil\vec{n}\rceil\lceil 1\rceil & \text{else} \end{cases} \\
&= \begin{cases} \lceil 2\rceil & \text{if} \quad G\lceil\vec{n}\rceil\lceil 2\rceil = \lceil 0\rceil \\ H\lceil\vec{n}\rceil\lceil 2\rceil & \text{else} \end{cases} \\
&= \ldots
\end{aligned}$$

The Double Fixed Point Theorem

$$\forall A \forall B \exists X \exists Y\, (X = AXY\ \ Y = BXY)$$

Proof.

Put

$$F \equiv \lambda x.[A(x\,true)(x\,false), B(x\,true)(x\,false)]$$

By the simple Fixed Point Theorem there exists a Z such that

$$FZ = Z$$

Take

$$X \equiv Z\,true, \quad Y \equiv Z\,false$$

Then

$$X = Z\,true = FZ\,true = A(Z\,true)(Z\,false) \equiv AXY$$

and similarly

$$Y = BXY.$$

Corollary.

Given contexts $C_i \equiv C_i[f, g, x], i = 1, 2$ there exist F_1, F_2

such that

$$F_1 x = C_1[F_1, F_2, x] \quad F_2 x = C_2[F_1, F_2, x]$$

Definition (coding λ-terms)

Let $\langle *, * \rangle : N^2 \to N$ be a recursive coding of ordered pairs of natural numbers. We put

(i) $v^{(0)} = v \qquad v^{(n+1)} = v^{(n)'}$

and similarly we define $c^{(n)}$.

(ii)

$$\# v^{(n)} = \langle 0, n \rangle \qquad \# c^{(n)} = \langle 1, n \rangle$$

$$\#(MN) = \langle 2, \langle \#M, \#N \rangle \rangle \qquad \#(\lambda x.M) = \langle 3, \langle \#x, \#M \rangle \rangle$$

We write

$$\ulcorner M \urcorner = \ulcorner \#M \urcorner$$

Theorem (Kleene)

There is an „interpreter" combinator E for closed λ-terms without constants such that for every closed term M without constants, we have

$$E \ulcorner M \urcorner = M.$$

Proof (P. de Bruin)

Construct E_1 such that for an arbitrary F

$$E_1 F \ulcorner x \urcorner = F \ulcorner x \urcorner$$
$$E_1 F \ulcorner MN \urcorner = (E_1 F \ulcorner M \urcorner)(E_1 F \ulcorner N \urcorner)$$
$$E_1 F \ulcorner \lambda x.M \urcorner = \lambda z.(E_1 F_{[x:=z]} \ulcorner M \urcorner)$$

where

$$F_{[x:=z]} \ulcorner n \urcorner = \begin{cases} F \ulcorner n \urcorner & \text{if} \quad \# n \neq x \\ z & \text{if} \quad \# n = x \end{cases}$$

Note that $F_{[x:=z]}$ can be written in the form

$$G \ulcorner x \urcorner z F.$$

By induction on M it follows that

$$E_1 F \ulcorner M \urcorner =$$
$$= M[x_1 := F \ulcorner x_1 \urcorner, \dots, x_n := F \ulcorner x_n \urcorner][c_1 := F \ulcorner c_1 \urcorner, \dots, c_m := F \ulcorner c_m \urcorner]$$

where

$\{x_1, \dots x_n\} = FV(M)$ $\quad \{c_1, \dots c_m\}$ are the constants in M.

Hence for closed M without constants, we have

$$E_1 I \ulcorner M \urcorner = M.$$

Now take

$$E \equiv \lambda x.E_1 I x = E_1 I.$$

Second Fixed Point Theorem

$$\forall F\, \exists X\; F\ulcorner X \urcorner = X$$

Proof.

By the recursiveness of #, there are recursive functions *AP* and *Num* such that

$$AP(\#M,\ \#N) = \#\, MN \qquad Num\,(n) = \#\ulcorner n \urcorner$$

Let *AP* and Num be λ-defined by closed terms ***AP*** and ***Num*** .

Then

$$\mathbf{AP}\ulcorner M \urcorner\ulcorner N \urcorner = \ulcorner MN \urcorner \qquad \mathbf{Num}\ulcorner n \urcorner = \ulcorner\ulcorner n \urcorner\urcorner$$

in particular, we have

$$\mathbf{Num}\ulcorner M \urcorner = \ulcorner\ulcorner M \urcorner\urcorner$$

If we put

$$W \equiv \lambda x.F(\mathbf{AP}x(\mathbf{Num}\,x)) \qquad X \equiv W\ulcorner W \urcorner$$

then

$$X \equiv W\ulcorner W \urcorner = F(\mathbf{AP}\ulcorner W \urcorner(\mathbf{Num}\ulcorner W \urcorner))$$
$$= F\ulcorner W\ulcorner W \urcorner\urcorner = F\ulcorner X \urcorner$$

The following result due to Scott is an application of the Second Fixed Point Theorem, which is useful in proving undecidability results. Its flavor resembles the well-known Rice's Theorem on recursively enumerable sets.

Theorem (Scott)

Let $A \subseteq \Lambda$ be a set of terms such that

(i) A is nontrivial, i.e. $A \neq 0$ and $A \neq \Lambda$

(ii) A is closed under $=$ that is

$$M \in A, M = N \Rightarrow N \in A$$

Then A is not recursive, more precisely, the set

$$\#A = \{\, \#M \mid M \in \Lambda \}$$

is not recursive.

Proof

by contradiction. Suppose that A is recursive. It follows that there is a closed λ-term F such that

$$M \in A \Leftrightarrow F\ulcorner M \urcorner = \ulcorner 0 \urcorner \quad M \notin A \Leftrightarrow F\ulcorner M \urcorner = \ulcorner 1 \urcorner$$

Take $M_0 \in A, M_1 \notin A$. Let

$$Gx = if\ Zero(Fx)\ then\ M_1\ else\ M_0$$

then

$$M \in A \Leftrightarrow G\ulcorner M \urcorner = M_1 \notin A$$

$$M \notin A \Leftrightarrow G\ulcorner M \urcorner = M_0 \in A$$

By the second Fixed Point Theorem, there is a term M such that

$$G\ulcorner M \urcorner = M$$

thus

$$M \in A \Leftrightarrow M = G\ulcorner M \urcorner \notin A$$

a contradiction.

Definition

(i) We say that a term M is in normal form if M has no part (redex) of the form

$$(\lambda x.P)Q$$

(ii) We say that a term M has a normal form if there is a term N in normal form such that $M = N$.

Example

I is in normal form, **IK** has a normal form.

Both types of numerals are in normal form etc.

If M is in normal form, no rule is applicable to M.

Note that the term

$$\Omega \equiv (\lambda x.xx)(\lambda x.xx)$$

has no normal form. The β-rule is applicable to the redex

$$(\lambda x.xx)(\lambda x.xx)$$

but the contractum remains Ω .

Theorem (Church, Scott)

The set

$$NF = \{ M \mid M \text{ a normal form}\}$$

is not recursive.

This result was first proved by Church (1936) by a different method. Historically it was the first example of noncomputable property.

Proof.

NF is closed under equality. We have shown that it is nonempty. We have noted that the term

$$\Omega \equiv (\lambda x.xx)(\lambda x.xx)$$

has no normal form. Thus $NF \neq \Lambda$ and hence it is nontrivial. The rest follows from the Church, Scott theorem.

The semantics of λ-calculus

- Semantics of a language L gives a „meaning" to the expressions in L. This can be given essentially in two ways.
- By providing a way in which expression of L are used. This gives so called *operational semantics of L.*
- By *translating* the expressions of L into expressions of another language that is already known. In this way we obtain a so called *denotational semantics of L.*

Operational semantics: reductions and strategies

There is a certain asymetry in the basic rule (β).

$$(\lambda x.x+1)3 = 3+1$$

The above equality an be interpreted as „3+1 is the result of computing $(\lambda x.x+1)3$", but not vice versa.

This computational aspect will be expressed by writing

$$(\lambda x.x+1)3 -\!\!>> 3+1$$

which reads „ $(\lambda x.x+1)3$ reduces to $3+1$ ".

Definition

A binary relation R on Λ is called

(i) *compatible* if it is compatible with the two operations of the λ-calculus

$$M\ R\ N \Rightarrow ZM\ R\ ZN$$

$$M\ R\ N \Rightarrow MZ\ R\ NZ$$

$$M\ R\ N \Rightarrow (\lambda x.M)\ R\ (\lambda x.N)$$

(ii) a *reduction* on Λ if it is a compatible reflexive and transitive relation.

(iii) a *congruence* on Λ if it is a compatible equivalence relation.

Definition

The binary relations $\to_\beta$, $\twoheadrightarrow_\beta$ and $=_\beta$ on Λ are defined as follows

(i) (1) $(\lambda x.M)N \to_\beta M[x := N]$

(2) $M \to_\beta N \Rightarrow ZM \to_\beta ZN$

$M \to_\beta N \Rightarrow MZ \to_\beta NZ$

$M \to_\beta N \Rightarrow (\lambda x.M) \to_\beta (\lambda x.N)$

(ii) (1) $M \twoheadrightarrow_\beta M$

(2) $M \to_\beta N \Rightarrow M \twoheadrightarrow_\beta N$

(3) $M \twoheadrightarrow_\beta N, N \twoheadrightarrow_\beta L \Rightarrow M \twoheadrightarrow_\beta L$

(iii) (1) $M \twoheadrightarrow_\beta N \Rightarrow M =_\beta N$

(2) $M =_\beta N \Rightarrow N =_\beta M$

(3) $M =_\beta N, N =_\beta L \Rightarrow M =_\beta L$

These relations are pronounced as follows

$M \to_\beta N$ reads „*M β-reduces to N in one step*“

$M \twoheadrightarrow_\beta N$ reads „ *M β-reduces to N* “

$M =_\beta N$ reads „*M is β-convertible to* N“

By definition we have

$\to_\beta$ is compatible

$\twoheadrightarrow_\beta$ is a reduction

$=_\beta$ is a congruence relation

Note that $\twoheadrightarrow_\beta$ is the reflexive transitive closure of $\to_\beta$

and $=_\beta$ is the equivalence relation generated by $\to_\beta$.

Proposition

$$M =_\beta N \Leftrightarrow \lambda \vdash M = N$$

Proof. $(\Rightarrow)$ By induction on the generation of $=_\beta$.

$(\Leftarrow)$ By induction on the length of proof.

Lemma.

Let M be a β-normal form. Then

$$M \twoheadrightarrow_\beta N \Rightarrow N \equiv M$$

Proof.

This is true if $\twoheadrightarrow_\beta$ is $\rightarrow_\beta$. The rest follows from transitivity.

The reduction is useful for an analysis of convertibility. The Church-Rosser Theorem states that if two terms are convertible, then they reduce to the same term.

In many cases it is possible to prove that two terms are not convertible by showing that they do not reduce to a common term.

Church-Rosser Theorem

If there are M, M_1, M_2 such that

$$\begin{tikzcd} & M \arrow[dl, "\beta"'] \arrow[dr, "\beta"] & \\ M_1 & & M_2 \end{tikzcd}$$

Then

$$\begin{tikzcd} & M \arrow[dl, "\beta"'] \arrow[dr, "\beta"] & \\ M_1 \arrow[dr, "\beta"'] & & M_2 \arrow[dl, "\beta"] \\ & N & \end{tikzcd}$$

for some λ-term N.

Corollary.

If $M_1 =_\beta M_2$ then there is a λ-term N such that $M_1 \twoheadrightarrow_\beta N$ and $M_2 \twoheadrightarrow_\beta N$.

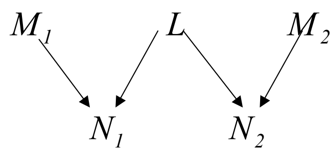
Proof.

By induction on the generation of $=_\beta$.

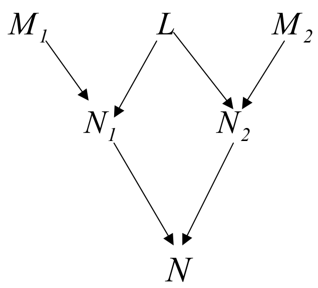
a) $M_1 =_\beta M_2$ because $M_1 \twoheadrightarrow_\beta M_2$. Take $N \equiv M_2$.

b) $M_1 =_\beta M_2$ because $M_2 =_\beta M_1$. Then by the induction hypothesis M_1 and M_2 have a common reduct N_1. Put $N \equiv N_1$.

c) $M_1 =_\beta M_2$ because $M_1 =_\beta L$, $L =_\beta M_2$. By the Induction hypothesis there are N_1, N_2 such that



It follows from the Church-Rosser Theorem that there is a common reduct N of N_1, N_2.



N is a common reduct of M_1, M_2.

Corollary.

(i) If N is a β-normal form of M then $M \twoheadrightarrow_\beta N$.

(ii) Every λ-term has at most one β-normal form.

Proof. (i) Let $M =_\beta N$ and N is in β-normal form. By the Corollary of the Church-Rosser Theorem M and N have a common reduct L. But this is equal to N.

(ii) Suppose that N_1, N_2 are two β-normal forms of M. Then $N_1 =_\beta N_2 =_\beta M$. Hence N_1, N_2 have a common reduct L. But then $N_1 \equiv L \equiv N_2$ since N_1, N_2

are in β-normal form.

Definition.

We say that a λ-calculus T is consistent if there are two terms of it such that $T \mid\!\not- M = N$. Otherwise T is inconsistent.

Theorem.

(i) λ-calculus is consistent.

Proof.

$\lambda \mid\!\not- true = false$ Otherwise $true =_\beta false$ and by Church Rosser Theorem it is impossible since $true$ and $false$ are distinct β-normal forms.

Note that $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$ has no β-normal form. Otherwise $\Omega \twoheadrightarrow_\beta N$ for some N in β-normal form. But Ω reduces to itself and is not in β-normal form.

Recall that the combinator $\boldsymbol{Y}$ finds fixed points

$$YF = F(YF)$$

On the other hand, we have

$$\begin{aligned} YF \equiv (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))F &\to_\beta \\ (\lambda x.(F(xx))(\lambda x.F(xx))) &\to_\beta \\ F((\lambda x.F(xx))(\lambda x.F(xx)) &\leftarrow_\beta \\ F((\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))F) &\equiv F(YF) \end{aligned}$$

Hence we do not have $YF \twoheadrightarrow_\beta F(YF)$ although this is often desirable.

Turing introduced another fixed point operator with the desired property.

Theorem (Turing´s fixed point combinator)

Let $\Theta = AA$ with $A = \lambda xy.y(xxy)$. Then for every F, we have

$$\Theta F \twoheadrightarrow F(\Theta F)$$

Proof.

$$\Theta F \equiv AAF \to F(AAF) \equiv F(\Theta F)$$

Similarly, one can find solutions for the double and for the second fixed point theorem that do reduce in an analogous manner.

Strategies

In order to find the β-normal form of a term M (if it exists), the various redexes can be reduced in different orders. In spite of this, the β-normal form is unique. However not every sequence of reductions leads to the (existing) β-normal form.

Example

$A \equiv K\mathbf{I}B$, with a term B without a β-normal form has a normal form $\mathbf{I}$ but A has an infinite reduction path by reducing within B e.g. $A \equiv K\mathbf{I}\Omega$.

A *reduction strategy* chooses one redex among the various possible redexes which can be reduced in the current step and thereby it determines how to reduce a term.

It turns out that there is a strategy that always normalizes terms that do have a β-normal form.

Definition. Leftmost strategy, Lazy strategy.

(i) The *main symbol* of a redex $(\lambda x.M)N$ is the first λ.

(ii) Let R_1, R_2 be two redexes that occur in a term *M*. We say that R_1 *is to the left of* R_2 if the main symbol of R_1 is to the left of that of R_2.

(iii) We write $M \to_l N$ if N results from M by contracting the leftmost redex in *M*. The reflexive transitive closure of $\to_l$ is denoted by $\twoheadrightarrow_l$.

The strategy that always contracts the leftmost redex is caled the *leftmost strategy* or the *normal strategy* and recently the *lazy strategy*. Computing in accord to the lazy strategy is called *lazy evaluation.*

The following theorem, due to Curry, states that if a term has a normal form then that normal form can be found by the lazy strategy.

Theorem. (Curry)

If M has a normal form N, then $M \twoheadrightarrow_l N$.

This reduction strategy is called lazy strategy because in an expression like

$$(\lambda ab.C[a,b])AB$$

it substitutes the subterms *A, B* directly into C[*a,b*] instead of evaluating them to normal forms.

Eager strategy performs a vice versa, it reduces first the subterms *A, B* to normal forms before substituting them into C[*a,b*] .

For the lambda calculus it is not possible to have an eager evaluation mechanism. This is due to the possibility of so-called *nonstrict functions* like

$$\lambda x.[0]$$

The *strict functions* are defined as follows: *F* is strict if for arbitrary $M_1, M_2, \ldots, M_n \in \Lambda$

$$FM_1M_2\ldots M_n = \perp$$

whenever for one of the M_i, $1 \leq i \leq n$, $M_i = \perp$ holds.

Note that the above defined function is nonstrict.

Nonstrict functions enhance the expressive power of the lambda calculus, but complicate the implementation of the language. That`s why the lambda calculus is is sometimes called a *lazy language.*
An almost eager evaluation is implemented in functional languages ML,SML and others, the lazy evaluation is implemented in Haskell.

Denotational semantics: set-theoretical models

Denotational semantics gives the meaning of a λ-term *M* by translating it to an expression denoting a set $\|M\|$. This set is an element of a mathematical structure in which application and abstraction are well-defined operations and the map $\|\cdot\|$ preserves these operations. In this way we obtain a so-called denotational semantics.

Constructing a model for the lambda calculus one would like to have a space D such that D is isomorphic to the space D^D. But this is impossible for cardinality reasons. In 1969 Scott solved this problem by restricting D^D to the continuous functions on D provided with a proper topology.

Scott worked with complete lattices with an induced topology and constructed a D such that $D^D \cong D$. It turned out that a model of the lambda calculus is obtained if D^D is a retract of D.

Definition.

A *complete lattice* is a partially ordered set $D = (D, \leq)$ such that for each $X \subseteq D$ the *supremum* $\mathbf{sup}\, X \in D$ exists.

Each D has a largest element *top* $T = \mathbf{sup}\, D$ and the least element *bottom* $\perp = \mathbf{sup}\, \emptyset$ and every $X \subseteq D$ has an infimum $\mathbf{inf}\, X = \mathbf{sup}\{y \mid \forall x \in X(y \leq x)\}$.

A subset $X \subseteq D$ is *directed* if $X \neq \emptyset$ and

$$\forall x, y \in X\, \exists z \in X[x \leq z \textit{ and } y \leq z].$$

Let $D, D', \ldots$ range over complete lattices.

Definition.

A mapping $f : D \to D'$ is *continuous* if for all directed $X \subseteq D$ one has

$$f(\mathbf{sup}\, X) = \mathbf{sup}\, f(X) = \mathbf{sup}\{f(x) \mid x \in X\}.$$

Note that each continuous function is monotoneous.

$$\begin{aligned} x \leq y \Rightarrow\ & y = \mathbf{sup}\{x, y\} \\ \Rightarrow\ & f(y) = f(\mathbf{sup}\{x, y\}) = \mathbf{sup}\{f(x), f(y)\} \\ \Rightarrow\ & f(x) \leq f(y). \end{aligned}$$

Definition. (product and lattice of continuous maps)

Let $D=(D,\leq), D'=(D',\leq')$.

(i) $D\times D'=\{(d,d')\mid d\in D, d'\in D'\}$ is the Cartesian product of D,D' ordered by

$$(d_1,d_1')\leq(d_2,d_2')\Leftrightarrow d_1\leq d_2 \text{ and } d_1'\leq' d_2'.$$

(ii) $[D\to D']=\{f:D\to D'\mid f \text{ is continuous }\}$ is a function space partially ordered by

$$f\leq g\Leftrightarrow \forall d(f(d)\leq' g(d)).$$

Lemma.

(i) $D\times D'$ is a complete lattice and for arbitrary $X\subseteq D\times D'$ we have

$$\mathbf{sup}\, X=(\mathbf{sup}(X)_0,\mathbf{sup}(X)_1),$$

where

$$(X)_0=\{d\in D\mid \exists d'\in D'\,(d,d')\in X\}$$
$$(X)_1=\{d'\in D'\mid \exists d\in D\,(d,d')\in X\}$$

(ii) $[D\to D']$ is a complete lattice if we apply pointwise convergence to continuous functions.
Namely, if $f_i:D\to D', i\in I$ is a collection of continuous maps and we define

$$f(x)=\mathbf{sup}_i(f_i(x))$$

then f is continuous and it is the supremum of the collection in $[D\to D']$.

Proof.

(i) Easy.

(ii) Let $X\subseteq D$ be directed then

$$\begin{aligned} f(\mathbf{sup}\, X) &= \mathbf{sup}_i\, f_i(\mathbf{sup}\, X) \\ &= \mathbf{sup}_i\, \mathbf{sup}_{x\in X}\, f_i(x) \quad \textit{continuity of } f_i \\ &= \mathbf{sup}_{x\in X}\, \mathbf{sup}_i\, f_i(x) \\ &= \mathbf{sup}_{x\in X}\, f(x). \end{aligned}$$

Thus f is continuous and $f=\mathbf{sup}\, f_i$ in $[D\to D']$.

If λ^s denotes the λ-abstraction in set theory, we have

$$\mathbf{sup}_i\, \lambda^s x.f_i(x)=\lambda^s x.\mathbf{sup}_i(f_i(x))$$

Hence **sup** commutes with λ^s.

Fixed Point Theorem.

Let $f\in[D\to D]$. Then f has a least fixed point defined by

$$Fix(f)=\mathbf{sup}_n\, f^n(\perp).$$

Note that the set $\{f^n(\bot) \mid n \in N\}$ is directed and

$\bot \leq f(\bot)$, hence by monotonicity, we get

$$\bot \leq f(\bot) \leq f^2(\bot) \leq \ldots$$

Therefore

$$f(Fix(f)) = \mathbf{sup}_n\, f(f^n(\bot)) = \mathbf{sup}_n\, f^{n+1}(\bot) = Fix(f).$$

If x is another fixed point of f then $f(x)=x$ and $\bot \leq x$,

thus by monotonicity $f^n(\bot) \leq f^n(x) = x$.

Hence $Fix(f) \leq x$.

Lemma on separate continuity.

Let $f : D \times D' \to D''$. Then f is continuous iff f is continuous in each of its variables separately i.e.

$\lambda^s x.f(x, x'_0)$ and $\lambda^s x'.f(x_0, x')$ are continuous for all x_0, x'_0.

Proof. $\Rightarrow$ as usual.

$\Leftarrow$ Let $X \subseteq D \times D'$ be directed. Then

$$\begin{aligned} f(\mathbf{sup}\, X) &= f(\mathbf{sup}(X)_0, \mathbf{sup}(X)_1) \\ &= \mathbf{sup}_{x \in (X)_0}\, f(x, \mathbf{sup}(X_1)) \\ &= \mathbf{sup}_{x \in (X)_0}\, \mathbf{sup}_{x' \in (X)_1}\, f(x, x') \\ &= \mathbf{sup}_{(x,x') \in X}\, f(x, x'). \end{aligned}$$

The last equality follows from the fact that X is directed.
Hence f is continuous.

We are going to define the lattice versions of operations of application and abstraction and we will show that these operations are continuous.

Definition.

Put

(i) $Ap : ([D \to D'] \times D) \to D'$ by $Ap(f, x) = f(x)$.

(ii) for $f \in [(D \times D') \to D'']$ define abstraction as follows

$$\lambda^s y.f(x, y).$$

Theorem.

(i) Ap is continuous.

(ii) $\lambda^s y.f(x, y) \in [D' \to D'']$ and depends continuously on x.

Proof.
We shall use the lemma on separate continuity.

(i) $\lambda^s x.Ap(f, x) = \lambda^s x.f(x) = f$ is continuous since $f \in [D \to D']$.

To prove the other continuity, let $f \in [D \to D']$.

put $H = \lambda^s f.Ap(f, x_0) = \lambda^s f.f(x_0)$. then for any directed

family $f_i, i \in I$, we have

$$\begin{aligned} H(\mathbf{sup}_i\, f_i) &= (\mathbf{sup}_i\, f_i)(x_0) \\ &= \mathbf{sup}_i (f_i(x_0)) \quad \text{by pointwise convergence} \\ &= \mathbf{sup}_i\, H(f_i) \end{aligned}$$

Hence Ap is continuous.

(ii) It follows from the separate continuity that

$$\lambda^s y.f(x,y) \in [D' \to D'']$$

Moreover, for a directed $X \subseteq D$ we have

$$\lambda^s y.f(\mathbf{sup}\, X, y) = \lambda^s y.\mathbf{sup}_x\, f(x,y)$$
$$= \mathbf{sup}_x\, \lambda^s y.f(x,y)$$

by continuity of f and the commutativity of supremum and set abstraction.

Definition.

(i) We say that D is a *retract* of D' and write $D < D'$, if there are continuous mappings F, G such that $F: D' \to D, G: D \to D'$ and $F \circ G = id_D$.

(ii) We say that D is *reflexive* if $[D \to D] < D$.

Remark. If $D < D'$ using maps F, G, then F is „onto“ and G is „one-to-one“. We may identify D with its image $G(D) \subseteq D'$. Then F „retracts“ the larger space D' to the subspace D.

We shall show that every reflexive complete lattice determines a model of the lambda calculus.

Definition.

Let D be reflexive due to mappings F, G. Hence

$$[D \to D] < D$$

$$F: D \to [D \to D] \subseteq D \qquad (1)$$

$$G: [D \to D] \to D \qquad (2)$$

(i) Thus for $x \in D$ we have $F(x) \in [D \to D]$. In this way elements of D become functions on D and we may write for application $x._F\, y = F(x)(y)$ $(\in D)$.

(ii) On the other hand, every function continuous on D becomes via G an element of D. Thus for continuous f, we may write (abstraction) $\lambda^G x.f(x) = G(f)$ $(\in D)$.

Definition.

A valuation in D is a map ρ which to every term variable x adds a value $\rho(x)$ in D.

Definition.

Let D be reflexive via F, G. Let ρ be a valuation in D and M be a λ-term. The *denotation* $\|M\|_\rho$ *of* M *in* D *under valuation* ρ is defined by induction on M as follows:

$$\|x\|^D_\rho = \rho(x)$$

$$\|PQ\|^D_\rho = \|P\|^D_\rho \cdot_F \|Q\|^D_\rho$$

$$\|\lambda x.P\|^D_\rho = \lambda^G d.\|P\|^D_{\rho(x:=d)}$$

where $\rho(x:=d)$ is the valuation ρ' with

$$\rho'(y) = \begin{cases} \rho(y) \text{ if } y \not\equiv x \\ d \quad \text{ if } y \equiv x \end{cases}$$

The definition is correct. By induction on P one can show the continuity of $\lambda^G d.\|P\|^D_{\rho(x:=d)}$.

Definition.

We say that $M=N$ *is true in* D and write $D \models M=N$ if for all valuations ρ, we have $\|M\|^D_\rho = \|N\|^D_\rho$.

Intuitively, the denotation $\|M\|^D_\rho$ is M interpreted in D where every lambda calculus application is interpreted as $\cdot_F$ and every abstraction λ as λ^G. For instance

$$\|\lambda x.xy\|^D_\rho = \lambda^G d.d\, \rho(y) = \lambda^G x.x\, \rho(y).$$

Notation.

If D is reflexive and ρ is a valuation, it is obvious that the denotation $\|M\|^D_\rho$ depends only on the values of ρ on $FV(M)$.

$$\rho \mid FV(M) = \rho' \mid FV(M) \Rightarrow \|M\|^D_\rho = \|M\|^D_{\rho'}$$

Where | denotes the function restriction. In particular for combinators, $\|M\|^D_\rho$ does not depend on ρ and may be written $\|M\|^D$. If D is clear from the context, we write $\|M\|_\rho$ or $\|M\|$.

Theorem.

If D is a reflexive complete lattice by means of mappings F and G, then D is a sound model for the lambda calculus. In other words, we have

$$\lambda \vdash M = N \Rightarrow D \models M = N.$$

Proof.

By induction of the proof of $M=N$. The only two interesting cases are the axiom (β) and the rule (ξ).

(β) is the scheme $(\lambda x.M)N = M[x:=N]$. For an arbitrary valuation ρ, we have

$$\|(\lambda x.M)N\|_\rho = (\lambda^G d.\|M\|_{\rho(x:=d)})\cdot_F \|N\|_\rho$$
$$= F(G(\lambda^s d.\|M\|_{\rho(x:=d)}))(\|N\|_\rho)$$
$$= (\lambda^s d.\|M\|_{\rho(x:=d)})(\|N\|_\rho) \qquad \{F\circ G = id_D\}$$
$$= \|M\|_{\rho(x:=\|N\|_\rho)}$$

We need

Lemma. $\|M[x:=N]\|_\rho = \|M\|_{\rho(x:=\|N\|_\rho)}$

Proof by induction on the structure of M. Write

$$P^* \equiv P[x:=N], \quad \rho^* = \rho(x:=\|N\|_\rho)$$

Then

$$\|x^*\|_\rho = \|N\|_\rho = \|x\|_{\rho^*}$$
$$\|y^*\|_\rho = \rho(y) = \|y\|_{\rho^*}$$
$$\|(PQ)^*\|_\rho = \|P^*\|_\rho \cdot_F \|Q^*\|_\rho$$
$$= \|P\|_{\rho^*} \cdot_F \|Q\|_{\rho^*} = \|(PQ)\|_{\rho^*} \qquad IH$$
$$\|(\lambda y.P)^*\|_\rho = \lambda^G d.\|P^*\|_{\rho(y:=d)} = \lambda^G d.\|P\|_{(\rho(y:=d))^*}$$
$$\|\lambda y.P\|_{\rho^*} = \lambda^G d.\|P\|_{\rho^*(y:=d)}$$

It suffices to note that $(\rho(y:=d))^* = \rho^*(y:=d)$.

We have proved $\|(\lambda x.M)N\|_\rho = \|M[x:=N]\|_\rho$ and the proof of the axiom (β) is complete.

The rule ξ: $M = N \Rightarrow \lambda x.M = \lambda x.N.$ We have to show

$$D \models M = N \Rightarrow D \models \lambda x.M = \lambda x.N.$$

Indeed

$$D \models M = N$$
$$\Rightarrow \|M\|_\rho = \|N\|_\rho \qquad \textit{for all } \rho$$
$$\Rightarrow \|M\|_{\rho(x:=d)} = \|N\|_{\rho(x:=d)} \qquad \textit{for all } \rho, d$$
$$\Rightarrow \lambda^s d.\|M\|_{\rho(x:=d)} = \lambda^s d.\|N\|_{\rho(x:=d)} \qquad \textit{for all } \rho$$
$$\Rightarrow \lambda^G d.\|M\|_{\rho(x:=d)} = \lambda^G d.\|N\|_{\rho(x:=d)} \qquad \textit{for all } \rho$$
$$\Rightarrow \|\lambda x.M\|_\rho = \|\lambda x.N\|_\rho \qquad \textit{for all } \rho$$
$$\Rightarrow D \models \lambda x.M = \lambda x.N$$

It remains to show that reflexive complete lattices do exist. We will give an example of a reflexive complete lattice called D_A. The method is due to Engeler and it is a code-free variant of the graph model Pω due to Plotkin and Scott.

Definition.

(i) Let A be a set, by induction on $n \in N$ define

$$B_0 = A$$
$$B_{n+1} = B_n \cup \{(\beta, b) \mid b \in B_n \text{ and } \beta \subseteq B_n, \beta \text{ finite}\}$$
$$B = \bigcup_n B_n$$
$$D_A = P(B) = \{x \mid x \subseteq B\}$$

D_A is considered as a complete lattice ordered by inclusion ($\subseteq$). The set B is the closure of A under the operation of forming ordered pairs. It is assumed that A consists of urelements in order it does not contain pairs $(\beta, b) \in B$.

(ii) Define

by $F : D_A \to [D_A \to D_A]$ $\quad G : [D_A \to D_A] \to D_A$

$$F(x)(y) = \{b \mid \exists \beta \subseteq y\ ((\beta, b) \in x)\}$$

$$G(f) = \{(\beta, b) \mid b \in f(\beta)\}.$$

We shall show later that F, G are continuos and prove the reflexivity. Let $f \in [D_A \to D_A]$, and $y \in D_A$ be arbitrary. We have

$$\begin{aligned} F \circ G(f)(y) &= F(\{(\beta, b) \mid b \in f(\beta)\})(y) \\ &= \{b \mid \exists \beta \subseteq y\ b \in f(\beta)\} \\ &= \bigcup_{\beta \subseteq y} f(\beta) \\ &= f(y) \end{aligned}$$

Since $y = \bigcup_{\beta \subseteq y} \beta$ is a directed supremum. We have

$$F \circ G = id_{[D_A \to D_A]}.$$

(a) F is continuous: let $X \subseteq D_A$ be directed.

$$\begin{aligned} F(\mathbf{sup}\, X)(y) = F(\bigcup X)(y) &= \{b \mid \exists \beta \subseteq y (\beta, b) \in \bigcup X\} \\ &= \bigcup_{x \in X} \{b \mid \exists \beta \subseteq y (\beta, b) \in x\} \\ &= \mathbf{sup}\, \{F(x)(y) \mid x \in X\} \end{aligned}$$

(b) continuity of G: let $Y \subseteq [D_A \to D_A]$ be directed, let $f = \mathbf{sup}\, Y$ hence $f(\beta) = \bigcup_{y \in Y} y(\beta)$.

Then $G(f) = \{(\beta, b) \mid b \in f(\beta)\} = \{(\beta, b) \mid b \in \bigcup_{y \in Y} y(\beta)\}$

$$\begin{aligned} &= \bigcup_{y \in Y} \{(\beta, b) \mid b \in y(\beta)\} \\ &= \mathbf{sup}\{G(y) \mid y \in Y\} \end{aligned}$$

Theorem. (Semantic proof of consistency of λ-calculus)

The lambda calculus is consistent: $\lambda \mid\!\not- true = false$.

Proof.

If $\lambda \mid\!- x = y$ then $D_A \models x = y$. It suffices to take a valuation ρ of variables in D_A such that $\rho(x) \neq \rho(y)$. Then $D_A \not\models x = y$ a contradiction.

Extending the language

Some language constructs in functional languages

(i) „*Let* $x = M$ *in* E“ stands for $(\lambda x.E)M$ or $E[x := M]$. The latter is usefull if we want to type expressions, the various expressions may need to be typed differently.

(ii) „*Letrecf* $\vec{x} = C[f, \vec{x}]$ *in* E“ stands for

Let $f = \Theta(\lambda f\,\vec{x}.C[f, \vec{x}])$ *in E.* Here Θ is the Turings fixed point operator.

Similarly one can define *Letrec* using the double fixed point.

Delta rules

are useful in extending the lambda calculus by „external“ functions. Implementations of functional languages exploit the standard arithmetics of the processor which is much more efficient than computations with numerals in the lambda calculus. Besides the type *integer,* they use the standard types *boolean* and *Char.*

To represent all this and more, we extend the lambda calculus by so-called δ-rules. They are very helpful in theoretical analysis of programs and proofs.

Motivation. One of the first versions of a δ-rule was used by Church (1941). He used the rule to test the equality of numerals. It is possible to formulate it in a more general setting.

Example.

Let X be a set of closed terms in normal form. For $M, N \in X$ we define

$$\delta MN \to \lambda xy.x \qquad if \quad M \equiv N$$
$$\delta MN \to \lambda xy.y \qquad if \quad M \not\equiv N$$

Note that this is not one contraction rule, but a *rule scheme.* For any two elements of X one contraction rule.

Both assumptions on terms *M, N* are necessary to keep the Church-Rosser property working.

Example.

(a) Put $M \equiv (\lambda x.x)(\lambda y.y)$

$N \equiv (\lambda z.z)$

then M is not in normal form, but both terms are without free variables. We have

$$\begin{tikzcd}
 & (\delta M)N \arrow[dl, "\delta"'] \arrow[dr, "\beta"] & \\
\lambda xy.y & & (\delta N)N \arrow[d, "\delta"] \\
 & & \lambda xy.x
\end{tikzcd}$$

(b) If we put $M \equiv (\lambda x.wx),\ N \equiv (\lambda x.x)$ then we have

$$\begin{tikzcd}[column sep=small]
 & & (\lambda w.((\delta M)N)(\lambda z.z)) \arrow[dl, "\beta"'] \arrow[dr, "\delta"] & \\
 & ((\delta(\lambda x.((\lambda z.z)x)))N) \arrow[d, "\beta"'] & & (\lambda w \lambda xy.y)(\lambda z.z) \arrow[d, "\beta"] \\
 & ((\delta(\lambda x.x))N) \arrow[dl, "\delta"'] & & \lambda xy.y \\
\lambda xy.x & & &
\end{tikzcd}$$

Definition.

Let $X \subseteq \Lambda$ be a set of closed terms in normal forms.
Usually we take constants for the elements of C, hence $X \subseteq C$.
Let $f : X^k \to \Lambda$ be an „externally defined" function. In order to represent f, a so-called δ-rules are added to the lambda calculus as follows:

(1) A special constant in C is selected and is given a name

$$\delta \quad (= \delta_f).$$

(2) New contraction rules are added to those of the lambda calculus:

$$\delta M_1 \ldots M_k \to f(M_1, \ldots M_k), \quad M_1, \ldots M_k \in X$$

For a given function f, this is not one contraction rule but in fact a *rule scheme.* The resulting extended calculus is called the λδ-calculus. The corresponding notions of reduction are denoted $\to_{\beta\delta}$, $\to\!\!>_{\beta\delta}$. So δ-reduction is not an absolute notion, but it depends on the choice of f.

Theorem. (Mitschke)

Let f be a function on a set of closed terms in normal form. Then the resulting notion of reduction $\to\!\!>_{\beta\delta}$ satisfies the Church-Rosser theorem.

The notion of normal form generalises to βδ-normal form. So does the concept of leftmost reduction. The $\beta\delta-$normal forms can be found by a leftmost reduction.

Theorem.

If $M \to\!\!>_{\beta\delta} N$ and N is in $\beta\delta-$normal form, then $M \to\!\!>_{l\beta\delta} N$.

Example. Set of δ-rules for the *booleans.*

The following constants are selected in C

$\underline{true}$, $\underline{false}$, $\underline{not}$, $\underline{and}$, $\underline{ite}$ (*for if then else*)

And the following δ-rules are introduced

$\underline{not}\ \underline{true} \to \underline{false}$ $\quad$ $\underline{not}\ \underline{false} \to \underline{true}$

$\underline{and}\ \underline{true}\ \underline{true} \to \underline{true}$ $\quad$ $\underline{and}\ \underline{true}\ \underline{false} \to \underline{false}$

$\underline{and}\ \underline{false} \to \underline{true} \to \underline{false}$ $\quad$ $\underline{and}\ \underline{false}\ \underline{false} \to \underline{false}$

$\underline{ite}\ \underline{true} \to \mathbf{true} \equiv \lambda xy.x$ $\quad$ $\underline{ite}\ \underline{false} \to \mathbf{false} \equiv \lambda xy.y$

It follows that

$$\underline{ite}\ \underline{true}\ x\ y \twoheadrightarrow x \qquad \underline{ite}\ \underline{false}\ x\ y \twoheadrightarrow y$$

Now we introduce some operations on the set of integers

$$\mathbf{Z} = \{\ldots -1, 0, 1, 2, \ldots\}.$$

Example. For each $n \in \mathbf{Z}$, we choose a constant in C and give it the name $\lceil n \rceil$. Moreover the following constants in C are selected

plus, minus, times, divide, error, equal

Then we introduce the following schemes of δ- rules. For $m, n \in \mathbf{Z}$,

$\mathbf{plus}\lceil m \rceil \lceil n \rceil \to \lceil m+n \rceil$, $\quad$ $\mathbf{minus}\lceil m \rceil \lceil n \rceil \to \lceil n-m \rceil$,

$\mathbf{times}\lceil m \rceil \lceil n \rceil \to \lceil m*n \rceil$

$\mathbf{divide}\lceil m \rceil \lceil n \rceil \to \lceil m:n \rceil$ *if* $m \neq 0$, $\quad$ $\mathbf{divide}\lceil m \rceil \lceil 0 \rceil \to \mathbf{error}$,

$\mathbf{equal}\lceil m \rceil \lceil m \rceil \to \underline{true}$, $\quad$ $\mathbf{equal}\lceil m \rceil \lceil n \rceil \to \underline{false}$ *if* $m \neq n$.

We may add rules like

$$\mathbf{plus}\lceil m \rceil \mathbf{error} \to \mathbf{error}$$

Exercise. Write down a λδ-term F, $F \twoheadrightarrow \lceil n!+n \rceil$.

Similar δ-rules can be introduced for the set of reals. Another set of δ-rules is concerned with characters.

Example. Let Σ be a linearly ordered alphabet. For each symbol $s \in \Sigma$ choose a constant $'s' \in C$. Moreover, we choose two constants $\delta_{\leq}, \delta_{=}$ in C and state the following δ-rules:

$\delta_{\leq}'s_1''s_2' \to \underline{true}$ if s_1 precedes s_2 Σ, the ordering of Σ,

$\delta_{\leq}'s_1''s_2' \to \underline{false}$ otherwise.

$\delta_{=}'s_1''s_2' \to \underline{true}$ if $s_1 = s_2$,

$\delta_{=}'s_1''s_2' \to \underline{false}$ otherwise.

In the lambda calculus, we have defined domains and the corresponding δ-rules of operations on the sets of standard data types

boolean *integer* *Char*

By this way, we made a first step to the theory of lambda calculi with types.