TYPED LAMBDA CALCULI

Andrzej S. Murawski
University of λeicester

http://www.cs.le.ac.uk/people/amurawski/mgs2011-tlc.pdf

THE COURSE

The aim of the course is to provide an introduction to the lambda calculus along with a selection of results on its operational and denotational semantics.

Rather like the chassis of a bus, which supports the vehicle but is unseen by its users, versions of λ or CL underpin several important logical systems and programming.

Cardone & Hindley

ONLINE RESOURCES

- R. Loader. Notes on Simply Typed Lambda Calculus (1998)

  `http://www.lfcs.inf.ed.ac.uk/reports/98/ECS-LFCS-98-381/`

- J.-Y. Girard. Proofs and Types. 1990

  `http://www.paultaylor.eu/stable/prot.pdf`

- H. Barendregt. Lambda Calculi with Types (1993)

  `ftp://ftp.cs.ru.nl/pub/CompMath.Found/HBK.ps`

- H. Barendregt. The impact of the lambda calculus in logic and computer science (1997)

  `ftp://ftp.cs.ru.nl/pub/CompMath.Found/church.ps`

- R. Cardone, J. R. Hindley. History of Lambda-Calculus and Combinatory Logic (2006)

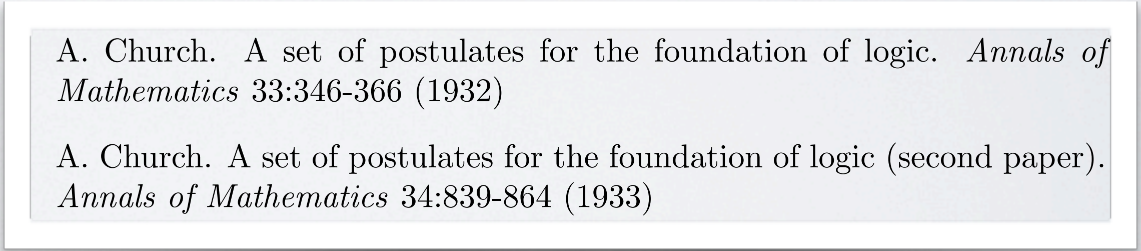
  `http://www-maths.swan.ac.uk/staff/jrh/papers/JRHHislamWeb.pdf`

SOME HISTORY

The lambda-calculus originated in order to study functions more carefully. Rosser

The untyped λ-calculus was originally part of a formal system proposed by Alonzo Church in 1928 with the intention of providing improved foundations of mathematics in turbulent times.

1903-1995

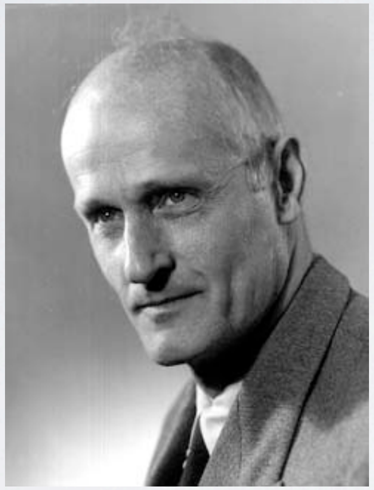

A. Church. A set of postulates for the foundation of logic. *Annals of Mathematics* 33:346-366 (1932)

A. Church. A set of postulates for the foundation of logic (second paper). *Annals of Mathematics* 34:839-864 (1933)

- Russell's paradox (1901)
- Gödel's incompleteness theorems (1931)

MORE HISTORY

Initially the **λ**-calculus was seen as a 'poor relative' and was not the primary focus of research in Church's group.


1909-1994

1907-1989

S. C. Kleene and J. B. Rosser. The inconsistency of certain formal logics. *Annals of Mathematics* 36: 630-636 (1935)

FUNCTION ABSTRACTION

$f(x) = x + 1$ $\lambda x(x + 1)$

The **λ**-notation turned out very successful.
Note that the function on the right is “anonymous”.

```
(lambda (x) (+ x 1))      fn x => x+1      fn x -> x+1

\x -> x+1     x => x+1     function (x) { return x+1; }

lambda x: x+1   [ :x | x+1 ]     Function(x) x+1
```

WHY LAMBDA?

- Class-abstraction [Russell and Whitehead, 1913]

$$\hat{x}(\phi(x))$$

- Function-abstraction [Church, 1932]

$$\wedge x(\phi(x))$$

- Ease of printing

$$\lambda x(\phi(x))$$

UNTYPED LAMBDA-TERMS

Let $\mathcal{V}$ be a countably infinite set of variables.

Variables	Any $x \in \mathcal{V}$ is a λ-term.
Function Application	If M, N are λ-terms, then (MN) is a λ-term.
Function Abstraction	If $x \in \mathcal{V}$ and M is a λ-term, then $(\lambda x.M)$ is a λ-term.

Notational conventions

It is common to omit brackets or λ's on the understanding that

- $M_1 M_2 M_3 \cdots M_n$ stands for $((\cdots((M_1 M_2) M_3) \cdots) M_n)$;
- $\lambda x.MN$ stands for $(\lambda x.(MN))$, not $((\lambda x.M)N)$;
- $\lambda x_1 x_2 \cdots x_n.M$ stands for $(\lambda x_1.(\lambda x_2. \cdots (\lambda x_n.M) \cdots))$.

SOME EXAMPLES

$$yx \qquad xx \qquad y(y(yx)) \qquad yyyx \qquad ((yy)y)x$$

$$\lambda f.\lambda x.fx \qquad \lambda fx.fx \qquad \lambda fx.f(fx) \qquad \lambda fx.f(f(fx))$$

$$(\lambda f.f(xx))(\lambda f.f(xx))$$

BINDING

- Like a quantifier in logic, λ is a *binding* construct with scope that extends over the body of the abstraction (indeed Church wrote $\lambda x(M)$ instead of $\lambda x.M$).

- An occurrence of x in M is *free* if it is not in scope of any λx. The free occurrences of x in $fxg(\lambda x.hzx)(hx)$ have been underlined below. Note that there can be occurrences of x that are not free (we call them *bound*).

$$f\underline{x}g(\lambda x.hzx)(h\underline{x})$$

- The outermost λx in $\lambda x.M$ binds only *free* occurrences of x in M.

FREE AND BOUND VARIABLES

A variable is *free* in M if it has at least one free occurrence in M. The set of *free* variables of a term can be defined as follows.

$$\begin{array}{rcl} fv(x) & = & \{x\} \\ fv(MN) & = & fv(M) \cup fv(N) \\ fv(\lambda x.M) & = & fv(M) \setminus \{x\} \end{array}$$

By analogy we can consider *bound* variables.

ALPHA-EQUIVALENCE

Names of bound variables are just a notational device to represent the binding structure within terms.

$$\lambda a.a \qquad \lambda b.b \qquad \lambda c.c \qquad \lambda d.d \qquad \lambda e.e \qquad \lambda f.f \qquad \lambda g.g \qquad \lambda h.h$$

$$\lambda x.fxg(\lambda x.hzx)(hx) \qquad \lambda x.fxg(\lambda y.hzy)(hx) \qquad \lambda y.fyg(\lambda x.hzx)(hy)$$

Different variable names can be used to convey the same information about binding. Then we talk of α-equivalent terms, written $M \equiv_\alpha N$.

Nominal set theory allows for an elegant formalization of α-equivalence!

ALPHA-EQUIVALENCE

Any term in the α-equivalence class of M can be obtained from M by a renaming of *bound* variables. The renaming must not modify the internal binding structure (the same occurrences of variables must be bound by the same occurrences of λ's).

$$\lambda x.\lambda x.x \equiv_\alpha \lambda x.\lambda y.y \not\equiv_\alpha \lambda x.\lambda y.x$$

$$\lambda x.f(\lambda y.y) \not\equiv_\alpha \lambda y.f(\lambda x.y)$$

$$\lambda f.f(\lambda x.f(\lambda y.y)) \not\equiv_\alpha \lambda f.f(\lambda x.f(\lambda y.x))$$

$$\lambda y.yx \not\equiv_\alpha \lambda x.xy \not\equiv_\alpha \lambda y.yy$$

DE BRUIJN INDICES

An alternative representation of binding: replace each bound occurrence of a variable with a numerical index that indicates the exact position of its binder relative to all potential binders (e.g. 1 means the innermost λ whose scope covers the relevant occurrence).

$$\lambda x.\lambda y.\lambda z.xz(yz) \qquad \lambda\lambda\lambda 31(21)$$

Two terms are α-equivalent if their de Bruijn representations are the same.

EQUIVALENCE CLASSES

- $\lambda x.\lambda x.x \equiv_\alpha \lambda x.\lambda y.y \not\equiv_\alpha \lambda x.\lambda y.x$

$$\lambda\lambda 1 \neq \lambda\lambda 2$$

- $\lambda f.f(\lambda x.f(\lambda y.y)) \not\equiv_\alpha \lambda f.f(\lambda x.f(\lambda y.x))$

$$\lambda 1(\lambda 2(\lambda 1)) \neq \lambda 1(\lambda 2(\lambda 2))$$

In what follows we shall identify terms with their α-equivalence class without using the explicit notation $[\cdots]_\alpha$. Care must then be taken to ensure that all definitions respect α-equivalence.

BETA RULE

Computation with λ-terms

$$(\lambda x.M)N \longrightarrow M[N/x]$$

Suppose substitution is defined simply as

$$\begin{array}{rcll} x[N/x] & = & N & \\ y[N/x] & = & y & x \neq y \\ (M_1 M_2)[N/x] & = & (M_1[N/x]\, M_2[N/x]) & \\ (\lambda y.M)[N/x] & = & (\lambda y.M[N/x]) & \end{array}$$

Then

$$((\lambda x.\lambda y.fxy)\, y)\, z \longrightarrow (\lambda y.fyy)z \longrightarrow fzz$$

During the substitution of y for x in $(\lambda y.fxy)$ the substituted occurrence y becomes *captured* by λy!

CAPTURE AVOIDANCE

$$\begin{array}{rcll} x[N/x] & = & N & \\ y[N/x] & = & y & x \neq y \\ (M_1 M_2)[N/x] & = & (M_1[N/x]\, M_2[N/x]) & \\ (\lambda y.M)[N/x] & = & (\lambda y.M[N/x]) & x \neq y,\ y \notin fv(N) \end{array}$$

Note that $(\lambda y.fxy)[y/x]$ is now undefined. But there exist terms M' such that $(\lambda y.fxy) \equiv_\alpha M'$ and $M'[y/x]$ is defined. For example

$$(\lambda z.fxz)[y/x] = (\lambda w.fyw).$$

This is no coincidence: whenever $M[N/x]$ is not defined, it will be possible to find $M' \equiv_\alpha M$ such that $M'[N/x]$ is defined. For instance, one can simply "refresh" the bound variables in M, i.e. rename them using variable names not occurring in M or N.

This is the notion of substitution we shall rely on in the course!

BETA-STEP

Basic β-step

$$(\lambda x.M)N \longrightarrow_b M'[N/x]$$

provided $M \equiv_\alpha M'$ and $M'[N/x]$ is defined. Note that $M'[N/x]$ is determined uniquely up to α-equivalence, so the above definition gives rise to a well-defined operation on α-equivalence classes of terms.

$$(\lambda x.\lambda y.fxy)y \longrightarrow_b \lambda w.fyw$$

$(\lambda x.M)N$ is called a β-redex.

ONE-STEP REDUCTION

We write $M \longrightarrow_{\beta 1} M'$ if M' is obtained from M by carrying out one basic β-step inside M (one-step β-reduction). Formally, $\longrightarrow_{\beta 1}$ is the smallest relation satisfying the following rules.

$$\frac{M \longrightarrow_{b} M'}{M \longrightarrow_{\beta 1} M'} \qquad \frac{M \longrightarrow_{\beta 1} M'}{\lambda x.M \longrightarrow_{\beta 1} \lambda x.M'}$$

$$\frac{M \longrightarrow_{\beta 1} M'}{MN \longrightarrow_{\beta 1} M'N} \qquad \frac{M \longrightarrow_{\beta 1} M'}{NM \longrightarrow_{\beta 1} NM'}$$

BETA-REDUCTION

- The reflexive and transitive closure of $\longrightarrow_{\beta 1}$, written $\longrightarrow^{*}_{\beta 1}$ or $\longrightarrow_{\beta}$ or $\xrightarrow{\beta}$, will be called β-**reduction**.

- The smallest equivalence relation containing $\longrightarrow_{\beta 1}$, written $=_{\beta}$, is called β-**equality** or β-**equivalence**.

TERMINOLOGY

A term M is **normal** (in normal form) if there is *no* N such that $M \longrightarrow_{\beta 1} N$.

A term M is **weakly normalizing** if there exists normal N such that $M \longrightarrow_{\beta} N$. We call N a **normal form** of M. If a term is not weakly normalizing, we write $M \Uparrow$.

A term M is **strongly normalizing** if there is no infinite sequence of terms $M_1, M_2, \cdots$ such that

$$M \longrightarrow_{\beta 1} M_1 \longrightarrow_{\beta 1} M_2 \longrightarrow_{\beta 1} \cdots .$$

CHURCH-ROSSER

If a term is weakly normalizing we can reduce it to some normal form. Are normal forms of terms determined uniquely (up to α-equivalence)? Is β-equality a consistent theory of λ-terms?

The two questions can be answered in the positive thanks to the following confluence property of $\longrightarrow_\beta$ (for α-equivalence classes of terms): whenever

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

there exists M' such that

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

UNIQUENESS OF NF

- Suppose N_1, N_2 are normal forms and

$$\begin{tikzcd} & M \arrow[dl, "\beta"'] \arrow[dr, "\beta"] & \\ N_1 & & N_2 \end{tikzcd}$$

  Then, by confluence, we must have $N_1 \equiv_\alpha N_2$.

- If we regard normal forms as values, the λ-calculus (equipped with β-reduction) can be viewed as a simple deterministic programming language. Not all computations yield values, though.

$$(\lambda x.xx)(\lambda x.xx) \longrightarrow_{\beta 1} (\lambda x.xx)(\lambda x.xx)$$

CONSISTENCY OF BETA

Recall that β-equivalence was defined the smallest equivalence relation containing $\longrightarrow_{\beta 1}$. More precisely, $M =_\beta N$ iff there exist $Q_1, \cdots, Q_k$ such that

$$M \quad \leftrightharpoons \quad Q_1 \quad \leftrightharpoons \quad \dots \quad \leftrightharpoons \quad Q_k \quad \leftrightharpoons \quad N$$

where $S \leftrightharpoons T$ stands for $S \longrightarrow_{\beta 1} T$ or $T \longrightarrow_{\beta 1} S$. Thanks to the Church-Rosser Theorem we can conclude that $M =_\beta N$ if and only if there exists Q such that

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

β-equivalence is consistent! For instance, $\lambda x.\lambda y.x \neq_\beta \lambda x.\lambda y.y$.

CHURCH NUMERALS

Natural numbers can be represented inside the λ-calculus using the idea of function iteration.

$$f \qquad \mapsto \qquad f^n = \underbrace{f \circ \ldots \circ f}_{n}$$

We define the term $\overline{n}$ representing the number n as follows.

$$\begin{array}{rcl}
\overline{0} & = & \lambda f.\lambda x.x \\
\overline{1} & = & \lambda f.\lambda x.fx \\
\overline{2} & = & \lambda f.\lambda x.f(fx) \\
 & \vdots & \\
\overline{n} & = & \lambda f.\lambda x.\underbrace{f(\cdots f(f}_{n}\,x)\cdots)
\end{array}$$

REPRESENTABILITY

One can use the above representation to represent numerical functions (even partial ones).

A term M is said to represent $f : \mathbb{N} \longrightarrow \mathbb{N}$ if

- $M\,\overline{n} \longrightarrow_\beta \overline{f(n)}$ for any $n \in \mathsf{dom}(f)$,
- $M\,\overline{n} \Uparrow$ for $n \notin \mathsf{dom}(f)$.

SOME ENCODINGS

successor	$\lambda n.\, \lambda f x.\, f(n f x)$
addition	$\lambda n_1 n_2.\, \lambda f x.\, (n_1 f)(n_2 f x)$
multiplication	$\lambda n_1 n_2.\, \lambda f x.\, n_1 (n_2 f) x$
exponentiation	$\lambda n_1 n_2.\, \lambda f x.\, n_1 n_2 f x$

CONDITIONAL

$$\textsf{if}\ \overline{0}\ x\ y \quad \longrightarrow_\beta \quad y$$

$$\textsf{if}\ \overline{n+1}\, x\ y \quad \longrightarrow_\beta \quad x$$

$$\lambda nxy.n(\lambda v.x)y$$

KLEENE'S PREDECESSOR

- Consider $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ defined by $g(x, y) = (x + 1, x)$.

  Observe that $g^n(0, 0) = (n, n - 1)$ for $n > 0$.

- To encode g in the λ-calculus we need to represent pairs.

$$\begin{array}{rcl} \langle M, N \rangle & = & \lambda z.zMN \\ \pi_1 & = & \lambda p.p(\lambda xy.x) \\ \pi_2 & = & \lambda p.p(\lambda xy.y) \end{array}$$

- Put it all together

$$\lambda n.\ \pi_2(n\, M_g\, \langle \overline{0}, \overline{0} \rangle)$$

CHURCH'S THESIS

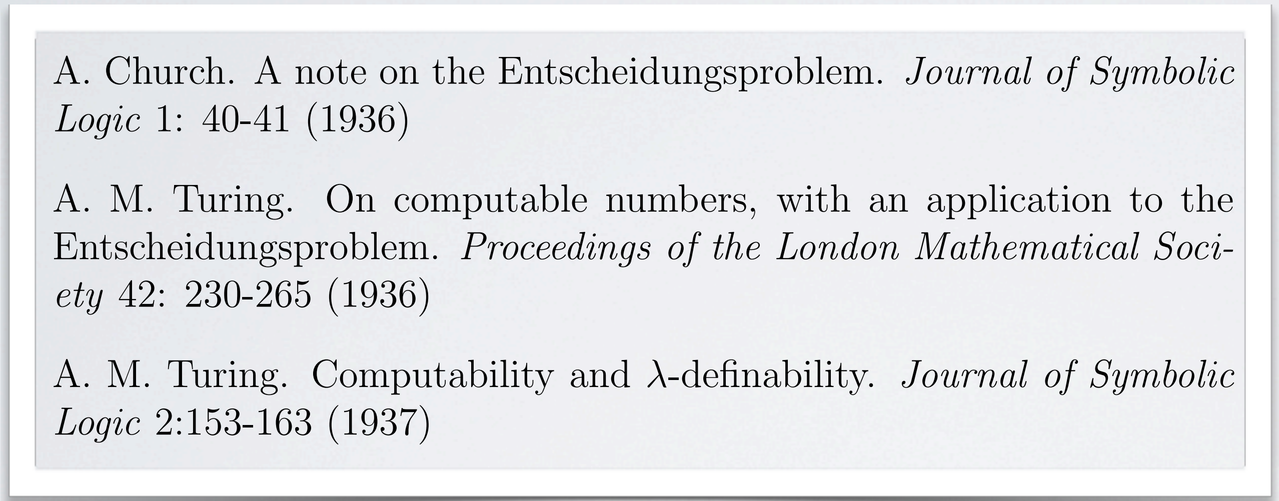
The λ-definable functions turned out to coincide with known classes of (recursive) functions.

A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics* 58:345–363 (1936)

S. C. Kleene. λ-definability and recursiveness. *Duke Mathematical Journal* 2:340–353 (1936)

This made Church conjecture that λ-definability captures the (informal) concept of "effective calculability". This is known as **Church's Thesis**.

ENTSCHEIDUNGPROBLEM



A. Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic* 1: 40-41 (1936)

A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* 42: 230-265 (1936)

A. M. Turing. Computability and λ-definability. *Journal of Symbolic Logic* 2:153-163 (1937)

1912-1954

FIXED POINTS

Let us consider the factorial function.

$$\mathsf{fact}(n) = \begin{cases} 1 & n = 0 \\ n \cdot \mathsf{fact}(n-1) & n > 0 \end{cases}$$

Note that fact is expressed in terms of fact.

Consider the function operator $G : (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N})$ defined as follows

$$G(f)(n) = \begin{cases} 1 & n = 0 \\ n \cdot f(n-1) & n > 0 \end{cases}$$

Observe that $G(\mathsf{fact}) = \mathsf{fact}$, i.e. fact is a fixed point.

Recursive definitions can be viewed as fixed points of operators on functions!

TURING'S COMBINATOR

Does any function have a fixed point in the λ-calculus? Given f, can we always find M such that

$$M \longrightarrow_\beta f(M)?$$

$$\begin{array}{rcl} \Omega & = & \lambda xy.y(xxy) \\ Y & = & \Omega\Omega \end{array}$$

Note the following

$$Yf = (\Omega\Omega)f \longrightarrow_\beta (\lambda y.y(\Omega\Omega y))f \longrightarrow_\beta f(\Omega\Omega f) = f(Yf)$$

Yf is the fixed point of f: $f(Yf) =_\beta Yf$. Y is a *fixed-point combinator.*

LAMBDA FACTORIAL

Recall that fact is a fixed point of $G : (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N})$.

$$G(f)(n) = \begin{cases} 1 & n = 0 \\ n \cdot f(n-1) & n > 0 \end{cases}$$

Consider the λ-term corresponding to G:

$$\lambda f.\, \lambda n.\, \mathsf{if}\ n\ (n \cdot f(n-1))\ \overline{1}.$$

Apply the fixed point combinator

$$Y(\lambda f.\, \lambda n.\, \mathsf{if}\ n\ (n \cdot f(n-1))\ \overline{1})$$

to obtain a term that represents the factorial function on Church numerals.

CURRY'S COMBINATOR

$$Y = \lambda f.\,(\lambda x.f(xx))\,(\lambda x.f(xx))$$

Observe that

$$Yf =_\beta f(Yf).$$

This time we do not have $Yf \longrightarrow_\beta f(Yf)$!

MGS COMBINATOR

$$T = \lambda abc \cdots xyz.z(midlands\, graduate\, school\, rulez)$$

$$Y = \underbrace{T \cdots T}_{26}$$

Observe that

$$Yf =_\beta f(Yf).$$

SCOTT NUMERALS

$$\begin{aligned}
\hat{0} &= \lambda xy.x \\
\hat{1} &= \lambda xy.y(\lambda xy.x) \\
\hat{2} &= \lambda xy.y(\lambda xy.y(\lambda xy.x)) \\
\hat{3} &= \lambda xy.y(\lambda xy.y(\lambda xy.y(\lambda xy.x)))
\end{aligned}$$

SOME ENCODINGS

successor	$\lambda n.\, \lambda xy.\, yn$
predecessor	$\lambda n.\, n\ *\ (\lambda x.x)$
conditional	$\lambda n.\, \lambda xy.\, n\, x\, y$

UNDECIDABLE PROBLEMS

- Is a given term weakly normalizing?
- Is a given term strongly normalizing?
- Does $M \longrightarrow_\beta N$ hold?
- Does $M =_\beta N$ hold?

EXTENSIONALITY

Extensionality is a fundamental property of functions.

$$\frac{Mx = Nx}{M = N} \quad x \notin fv(M) \cup fv(N)$$

But it is not reflected in the current system (based on β-equality alone). Observe that the following rule cannot be derived in general.

$$\frac{Mx =_\beta Nx}{M =_\beta N} \quad x \notin fv(M) \cup fv(N)$$

Can you see cases in which it can?

To incorporate extensionality, one could admit the rule given above or, equivalently, add the η-law.

$$\frac{}{\lambda x.Mx = M} \quad x \notin fv(M)$$

ETA-REDUCTION

Basic η-step ($x \notin fv(M)$):

$$\lambda x.Mx \longrightarrow_e M$$

Like for $\longrightarrow_b$, we can define:

- a one-step $\beta\eta$-reduction ($\longrightarrow_{\beta\eta 1}$) by allowing $\longrightarrow_b$ and $\longrightarrow_e$ inside terms and not only at the outermost level;
- $\beta\eta$-reduction ($\longrightarrow_{\beta\eta}$) as the symmetric and transitive closure of $\longrightarrow_{\beta\eta 1}$;
- $\beta\eta$-equality ($=_{\beta\eta}$) as the smallest equivalence relation containing $\longrightarrow_{\beta\eta 1}$.

$=_{\beta\eta}$ satisfies the *extensionality principle.* Suppose $Mx =_{\beta\eta} Nx$, where $x \notin fv(M) \cup fv(N)$. Then $\lambda x.Mx =_{\beta\eta} \lambda x.Nx$. Because $\lambda x.Mx =_{\beta\eta} M$ and $\lambda x.Nx =_{\beta\eta} N$, we can conclude $M =_{\beta\eta} N$.

FURTHER LAWS?

How about adding the following law?

$$\lambda xy.x =_{law} \lambda xy.y$$

Equating any different $\beta\eta$-normal forms M, N leads to inconsistency!

Böhm's Theorem: There exists a context C such that

$$C[M] =_{\beta\eta} x \qquad\qquad C[N] =_{\beta\eta} y.$$

A closed ($fv(M) = \emptyset$) term M has a head normal form (is *solvable*) if $M =_{\beta} \lambda x_1 \cdots x_n.x_i \cdots$. If a term has a normal form, then it has a head normal form. Equating unsolvable terms does not lead to inconsistency.

TYPES

In the untyped λ-calculus application is unconstrained. Terms can even be applied to themselves! This flexibility turns out to be a source of computational expressiveness. In what follows we shall impose constraints on the use of application, using types.

Let $\mathcal{G}$ be the set of **ground** types. For the time being, let $\mathcal{G} = \{o\}$.

- Any $A \in \mathcal{G}$ is a type.
- If A and B are types, so is $A \to B$.

TYPED LAMBDA CALCULUS

A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic* 5: 56-68 (1934)

H. B. Curry. Functionality in combinatory logic. *Proceedings of the National Academy of Science* 20: 584-590 (1934)

PRE-TERMS

In order define *typed* terms, we slightly modify the previous definition of λ-terms and add type annotations to λ-abstractions.

Variables	Any $x \in \mathcal{V}$ is a pre-term.
Function Application	If M, N are pre-terms, then (MN) is a pre-term.
Function Abstraction	If $x \in \mathcal{V}$, M is a pre-term, A is a type then $(\lambda x^A.M)$ is a pre-term.

As before we shall identify α-equivalent pre-terms, i.e. we consider α-equivalence classes (even though we shall not write $[\cdots]_\alpha$ explicitly).

Pre-terms are candidates for typed terms. Next we define when a pre-term can become a *typed* term. This will be the case if it can pass a typing test.

TYPING

Typing judgments have the form

$$x_1 : A_1, \cdots, x_n : A_n \vdash M : A$$

where

- $x_1, \cdots, x_n \in \mathcal{V}$
- $A_1, \cdots, A_n, A$ are types
- M is a pre-term.

One can think of the left-hand-side of the judgment (called the **context**) as a typing declaration for variables that might occur freely in M.

Indeed, the following invariant will be maintained: $fv(M) \subseteq \{x_1, \cdots, x_n\}$.

TYPING RULES

$$\frac{}{x_1 : A_1, \cdots, x_n : A_n \vdash x_i : A_i} \quad 1 \leq i \leq n$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A . M : A \to B}$$

PROPERTIES

Uniqueness: If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$ then $A = B$.

Weakening: If $\Gamma \vdash M : A$, $y \notin \Gamma$ and B is a type then $\Gamma, y : B \vdash M : A$.

Substitution: If $\Gamma \vdash N : B$ and $\Gamma, x : B \vdash M : A$ then $\Gamma \vdash M[N/x] : A$.

Subject Reduction: If $\Gamma \vdash M : A$ and $M \longrightarrow_{\beta\eta} N$ then $\Gamma \vdash N : A$.

Type inference I: There exists an algorithm that, given Γ and pre-term M, decides whether there exists A such that $\Gamma \vdash M : A$ and returns such A.

Type inference II: There exists an algorithm that, given an untyped term M, decides whether M can be annotated with types so that for the resultant pre-term M' there exist Γ and A such that $\Gamma \vdash M' : A$. There are ways to infer the most general such A (principal type).

WEAK NORMALIZABILITY

In the untyped λ-calculus it is undecidable whether a term is weakly normalizing, i.e. whether it reduces to a normal term. This changes dramatically with types: all terms have a normal form.

Some useful definitions.

- The degree of a type is defined by

$$\begin{array}{rcl} d(o) & = & 0 \\ d(A \to B) & = & 1 + \max(d(A), d(B)) \end{array}$$

- The degree of a redex $(\lambda x^A.M)N$ is the degree of the type of $\lambda x^A.M$.
- The degree of a term is equal to the highest degree of a redex occurring in it.

A NORMALIZATION ROUTINE

Given a term $\Gamma \vdash M$ proceed as follows.

- Let d be the degree of M.
- Consider all redexes of degree d in M and pick from among these a redex
$$\cdots(\lambda x^A.M')N'\cdots$$
such that all redexes in M' and N' have degree less than d, i.e. innermost redex of the highest degree.
- Fire it!

WHAT'S HAPPENED

$$\cdots(\lambda x^A.M')N'\cdots$$

- Thanks to our choice (innermost), there is now one fewer redex of degree d.
- There may be new redexes, but their degree will be that of the type of M' or N', both of which are strictly smaller than d.
- The procedure can be iterated until there are no redexes of degree d left.
- The whole routine can be repeated for the resultant term. Note that the degree of the term will now be strictly smaller than d. This implies termination.

ONE-STEP BOUNDS

The length of a term is defined as follows.

$$\begin{array}{rcl} l(x) & = & 1 \\ l(MN) & = & 1 + l(M) + l(N) \\ l(\lambda x.M) & = & 1 + l(M) \end{array}$$

Let M be a term whose degree is d. There exists a term M' of degree strictly smaller than d such that:

- $M \longrightarrow_\beta M'$ in at most $l(M)$ steps,
- $l(M') \leq 2^{l(M)}$.

UPPER BOUNDS

Let 2_n^m represent the following tower of exponentials

$$2^{2^{\cdot^{\cdot^{2^m}}}}$$

with n occurrences of 2.

Suppose M has degree d.

1. Its normal form has length $2_d^{l(M)}$.
2. Its normal form can be reached in $2_{d-1}^{l(M)}$ steps.

These bounds can be matched. β-reduction is not elementary (Statman)!

STRONG NORMALIZABILITY

Weak normalizability was initially regarded as more interesting, as it was all one needed to demonstrate consistency. In the context of computers, strong normalizability becomes a safety guarantee.

In the untyped λ-calculus it is undecidable whether a term is strongly normalizing. This also changes with types: all typed terms are strongly normalizing. It is not so easy to come up with a proof of this fact.

Exercise: Try a naive inductive proof to see what happens!

REDUCIBILITY (TAIT)

We define reducibility sets Red_A as follows.

$$\begin{array}{lllll}
\mathsf{Red}_o & = & \{ & (\Gamma, M) & | & \Gamma \vdash M : o, \quad M \text{ is strongly normalizing}\} \\
\mathsf{Red}_{A \to B} & = & \{ & (\Gamma, M) & | & \Gamma \vdash M : A \to B, \\
 & & & & & (\Gamma \cup \Delta, MN) \in \mathsf{Red}_B \text{ for all } (\Gamma \cup \Delta, N) \in \mathsf{Red}_A \quad \}
\end{array}$$

Then it is not hard to show that

- If $(\Gamma, M) \in \mathsf{Red}_A$ then M is strongly normalizing.
- For all Γ, if $(x : A) \in \Gamma$ then $(\Gamma, x) \in \mathsf{Red}_A$.

STRONG NORMALIZABILITY

By induction on the structure of M one can show the following.

Given $x_1 : A_1, \cdots, x_n : A_n \vdash M : A$, if $(\Gamma, N_i) \in \mathsf{Red}_{A_i}$ for any $i = 1, \cdots, n$ then $(\Gamma, M[N_1/x_1, \cdots, N_n/x_n]) \in \mathsf{Red}_A$.

By setting $N_i = x_i$ we finally arrive at:

$$\text{if } \Gamma \vdash M : A \text{ then } (\Gamma, M) \in \mathsf{Red}_A.$$

Since being in Red_A implies strong normalizability (previous slide), we can conclude that any term is strongly normalizing.

REPRESENTABILITY

Church numerals can be typed. Let us write $\overline{n}$ for $\lambda f^{o \to o}.\lambda x^{o}.f^{n}(x)$ and observe that

$$\vdash \overline{n} : (o \to o) \to (o \to o).$$

Let us write N for $(o \to o) \to (o \to o)$. As in the untyped case, we can define a notion of representability for (total) numeric functions:

$$\vdash M : \underbrace{\mathsf{N} \to \cdots \to \mathsf{N}}_{k} \to \mathsf{N}$$

represents $f : \mathbb{N}^k \to \mathbb{N}$ iff

$$M\overline{n_1} \cdots \overline{n_k} =_{\beta\eta} \overline{f(n_1, \cdots, n_k)}$$

for all $(n_1, \cdots, n_k) \in \mathbb{N}^k$. In the untyped case we could represent all recursive functions in the same way. Now only the *extended polynomials* can be represented (Schwichtenberg).

EXTENDED POLYNOMIALS

Let us fix a set $\mathcal{V}_{\mathbb{N}} = \{n_1, n_2, \cdots\}$ of (numeric) variables (for constructing polynomials). The set of extended polynomials is the smallest set $\mathcal{E}$ satisfying the rules below.

$$\frac{}{0, 1 \in \mathcal{E}} \qquad \frac{n \in \mathcal{V}_{\mathbb{N}}}{n \in \mathcal{E}} \qquad \frac{P_1, P_2 \in \mathcal{E}}{(P_1 + P_2) \in \mathcal{E}} \qquad \frac{P_1, P_2 \in \mathcal{E}}{(P_1 \cdot P_2) \in \mathcal{E}} \qquad \frac{P_1, P_2, P_3 \in \mathcal{E}}{\mathsf{ifzero}(P_1, P_2, P_3) \in \mathcal{E}}$$

The ifzero function is defined by: $\mathsf{ifzero}(m, m_0, m_1) = \begin{cases} m_0 & m = 0 \\ m_1 & m > 0 \end{cases}$

Examples

1. $(n_1 + (n_1 \cdot n_2))$
2. $(n_1 \cdot \mathsf{ifzero}(n_1 + n_2,\ 0,\ n_3 \cdot (1 + 1)))$

ALTERNATIVES

The previous notion uses the type N for representing both the arguments and the result. A more general notion is possible. Let

$$\mathsf{N}_A = (A \to A) \to (A \to A)$$

so that $\mathsf{N}_o = \mathsf{N}$. We have $\vdash \lambda f^{A\to A}.\lambda x^A.f^n(x) : \mathsf{N}_A$.

One can use different A's for arguments and the result. This turns out to *extend* the class of functions that can be represented, e.g. predecessor and exponentiation become representable. Equality and substraction are not, though.

S. Fortune, D. Leivant, M. O'Donnell. The Expressiveness of Simple and Second-Order Type Structures. *Journal of the ACM* 30(1): 151-185 (1983)

LOWER BOUNDS

$$\overline{n}_A = \lambda f^{A \to A} x^A . f^n x : \mathsf{N}_A$$

Note that

$$\overline{m}_{A \to A}\, \overline{n}_A =_{\beta\eta} (\overline{n^m})_A$$

Consider the big terms:

$$\begin{array}{rcl} \mathsf{big}_{A,0} & = & \overline{0}_A \\ \mathsf{big}_{A,n+1} & = & \mathsf{big}_{A \to A,n}\, \overline{2}_A \end{array}$$

Then $\mathsf{big}_{o,n}$ is of degree $n+2$, length linear in n, and

$$\mathsf{big}_{A,n} \longrightarrow_\beta (\overline{2^0_n})_A .$$

SOME HARD PROBLEMS

Many undecidable problems were associated with the untyped λ-calculus. Here are some problems that are have proved hard to solve in the typed case.

Unification: Suppose $x_1, \cdots, x_n \vdash M, N$. Do there exist terms $Q_1, \cdots, Q_n$ such that $M[Q_1/x_1, \cdots, Q_n/x_n] =_{\beta\eta} N[Q_1/x_1, \cdots, Q_n/x_n]$?

Matching: Same as above except that N must be closed.

Unification is undecidable (Goldfarb). Matching (for $\mathcal{G} = \{o\}$) is decidable (Stirling). Curiously, it is undecidable when $=_{\beta\eta}$ is replaced with $=_\beta$ (Loader).

MODELS

The typed λ-calculus can be given a natural set-theoretic interpretation, consistent with our intuition about functions.

- Each type A will be interpreted by a set, written $[\![A]\!]$.
- Each term-in-context $\Gamma \vdash M : A$ will be interpreted by a function, written $[\![\Gamma \vdash M : A]\!]$. Let $\Gamma = \{x_1 : A_1, \cdots, x_n : A_n\}$.
  - The domain of the function will be $[\![A_1]\!] \times \cdots \times [\![A_n]\!]$.
  - Its codomain will be $[\![A]\!]$.

  $\times$ stands for the Cartesian product of sets. Note that a closed term $\vdash M : A$ will be interpreted by a function in $1 \to [\![A]\!]$, i.e. an element of $[\![A]\!]$.

SET-THEORETIC MODEL

To get off the ground with the interpretation, we need an assignment of sets to all ground types, e.g. $[\![o]\!] = \{0, 1, 2, 3\}$. Starting from sets $[\![A]\!]$ for any $A \in \mathcal{G}$, we can interpret the remaning types inductively by

$$[\![A \to B]\!] = [\![A]\!] \Rightarrow [\![B]\!],$$

where $\Rightarrow$ stands for the set-theoretic function space, i.e. the set of all functions from $[\![A]\!]$ to $[\![B]\!]$.

Let $\Gamma = \{x_1 : A_1, \cdots, x_n : A_n\}$, $(a_1, \cdots, a_n) \in [\![A_1]\!] \times \cdots \times [\![A_n]\!]$ and $a \in [\![A]\!]$. The interpretation of terms is defined by induction on the structure of their typing derivations as follows.

- $[\![x_1 : A_1, \cdots, x_n : A_n \vdash x_i : A_i]\!]\,(a_1, \cdots, a_n) = a_i$
- $[\![\Gamma \vdash MN]\!]\,(a_1, \cdots, a_n) = (\ [\![\Gamma \vdash M]\!](a_1, \cdots, a_n)\ )(\ [\![\Gamma \vdash N]\!](a_1, \cdots, a_n)\ )$
- $(\ [\![\Gamma \vdash \lambda x^A.M]\!]\,(a_1, \cdots, a_n)\ )\,(a) = [\![\Gamma, x : A \vdash M]\!]\,(a_1, \cdots, a_n, a)$

SOUNDNESS

If $\Gamma \vdash M, N : B$ and $M =_{\beta\eta} N$, then $[\![\Gamma \vdash M : B]\!] = [\![\Gamma \vdash N : B]\!]$.

How about the converse? Consider the following cases.

1. $[\![o]\!] = \{\star\}$
2. $[\![o]\!] = \{0, \cdots, n\}$
3. $[\![o]\!] = \mathbb{N}$

Different Church numerals are not $\beta\eta$-equivalent, so for the converse to hold $[\![\mathsf{N}]\!]$ must be infinite.

COMPLETENESS

We write $[\![\Gamma \vdash M]\!]_X$ for $[\![\Gamma \vdash M]\!]$ obtained by setting $[\![o]\!] = X$.

Friedman: If $[\![\Gamma \vdash M]\!]_{\mathbb{N}} = [\![\Gamma \vdash N]\!]_{\mathbb{N}}$ then $M =_{\beta\eta} N$.

Plotkin: If $[\![\Gamma \vdash M]\!]_{\{0,\cdots,n\}} = [\![\Gamma \vdash N]\!]_{\{0,\cdots,n\}}$ for all $n \in \mathbb{N}$, then $M =_{\beta\eta} N$.

DEFINABILITY

If $[\![o]\!]$ is finite, so is $[\![A]\!]$ for any type.

- Suppose $[\![o]\!] = \{0,1\}$. Then the set $[\![o \to o]\!] = \{0,1\} \Rightarrow \{0,1\}$ has four elements. What can $[\![\vdash M : o \to o]\!]$ be?
- The only normal form (with respect to $\beta\eta$) at this type is $\lambda x^o.x$, which is interpreted by the identity function.
- The other three elements are not λ-*definable*, i.e. they are not interpretations of any terms.

In general it is undecidable whether a function is λ-definable, a surprising result due to Loader.

CARTESIAN-CLOSED CAT'S

The set-theoretic interpretation and the associated soundness result are instances of a more general category-theoretic interpretation in Cartesian Closed Categories, i.e. categories with products and function spaces. The typed λ-calculus can itself be organized into such a category.

- Objects are types.
- Morphisms are terms.

  A morphism between A and B is a $=_{\beta\eta}$-equivalence class of $x : A \vdash M : B$. The identity morphism is defined by $x : A \vdash x : A$.

- Composition is substitution.

Any (standard) interpretation of the typed lambda calculus in a cartesian closed category can be factorized through the syntactic interpretation above.

CURRY-HOWARD

Let us consider a few selected terms and their types.

Term	Type
$\lambda x^A.x$	$A \to A$
$\lambda x^A.\lambda y^B.x$	$A \to (B \to A)$
$\lambda x^{A\to(B\to C)}.\lambda y^{A\to B}.\lambda z^A.xz(yz)$	$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$

Think of $\to$ as implication in propositional logic. Are the types tautologies?

LOGIC OF TYPING

Let us revisit the typing rules, but with terms erased.

$$\frac{}{A_1, \cdots, A_n \vdash A_i} \quad 1 \leq i \leq n$$

$$\frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$$

The rules turn out to correspond to provability in (implication-only) *intuitionistic logic*, which is a proper subset of classical logic.

CORRESPONDENCE

LOGIC	**PROGRAMMING**
intuitionistic logic	λ-calculus
formulas	types
proofs	terms
simplification	reduction
provability	inhabitation

Can be useful in both directions!

PEIRCE'S LAW

$A = ((\alpha \to \beta) \to \alpha) \to \alpha$ is provable in classical logic. Let us see through the lens of the λ-calculus whether it is provable in intuitionistic logic. For a change we need to set $\mathcal{G} = \{\alpha, \beta\}$.

- Assume that A is provable in intuitionistic logic. Then there exists a term M such that $\vdash M : A$. Hence, there exists a $\beta\eta$-normal term M such that $\vdash M : A$.
- M must have the shape $\lambda x^{(\alpha\to\beta)\to\alpha}.N$, i.e. $x : (\alpha \to \beta) \to \alpha \vdash N : \alpha$. N must then be of the form xQ, where $x \vdash Q : \alpha \to \beta$. Thus, Q must have the shape $\lambda y^{\alpha}.R$ and

  $$x : (\alpha \to \beta) \to \alpha,\, y : \alpha \vdash R : \beta.$$

  Is this possible with R in normal form?

APPLIED LAMBDA CALCULI

We saw that the internal notion of representability in the typed λ-calculus is rather weak. But, of course, we would like to have typed languages that are Turing-strong. What can we do?

Types: introduce “meaningful” ground types

Terms: allow ground values and primitive operations as constants

Reduction: add suitable reduction rules

PCF

Ground types

$$\mathcal{G} = \{\mathbf{nat}\}$$

Constants

$$\frac{n \in \mathbb{N}}{\vdash n : \mathbf{nat}} \qquad \frac{}{\vdash \mathsf{succ} : \mathbf{nat} \to \mathbf{nat}} \qquad \frac{}{\vdash \mathsf{pred} : \mathbf{nat} \to \mathbf{nat}}$$

$$\frac{}{\vdash \mathsf{ifzero}_A : \mathbf{nat} \to (A \to (A \to A))} \qquad \frac{}{\vdash \mathsf{Y}_A : (A \to A) \to A}$$

BETA RECAP

Basic rule

$$(\lambda x.M)N \longrightarrow M[N/x]$$

Contextual rules

$$\frac{M \longrightarrow M'}{\lambda x.M \longrightarrow \lambda x.M'} \qquad \frac{M \longrightarrow M'}{MN \longrightarrow M'N} \qquad \frac{M \longrightarrow M'}{NM \longrightarrow NM'}$$

A basic step can be made anywhere inside the term (this will be restricted in what follows in favour of the leftmost-outermost strategy).

CALL-BY-NAME EVALUATION

Basic rules

$$\begin{array}{rcl} \mathsf{pred}\ 0 & \longrightarrow & 0 \\ \mathsf{pred}\ (n+1) & \longrightarrow & n \\ \mathsf{succ}\ n & \longrightarrow & n+1 \end{array}$$

$$\begin{array}{rcl} \mathsf{ifzero}_A\ 0\ M_0\ M_1 & \longrightarrow & M_0 \\ \mathsf{ifzero}_A\ (n+1)\ M_0\ M_1 & \longrightarrow & M_1 \end{array}$$

$$\begin{array}{rcl} (\lambda x.M)N & \longrightarrow & M[N/x] \\ \mathsf{Y}_A M & \longrightarrow & M(\mathsf{Y}_A M) \end{array}$$

Contextual rules

$$\frac{M \longrightarrow M'}{\mathsf{pred}\, M \longrightarrow \mathsf{pred}\, M'} \qquad \frac{M \longrightarrow M'}{\mathsf{succ}\, M \longrightarrow \mathsf{succ}\, M'} \qquad \frac{M \longrightarrow M'}{\mathsf{ifzero}_A\, M \longrightarrow \mathsf{ifzero}_A\, M'}$$

$$\frac{M \longrightarrow M'}{MN \longrightarrow M'N}$$

CALL-BY-VALUE EVALUATION

Let V stand for n, $\lambda x.M$ or $\mathsf{Y}_{A\to B}$.

Basic rules

$$\begin{array}{rcl} \mathsf{pred}\ 0 & \longrightarrow & 0 \\ \mathsf{pred}\ (n+1) & \longrightarrow & n \\ \mathsf{succ}\ n & \longrightarrow & n+1 \end{array} \qquad \begin{array}{rcl} \mathsf{ifzero}_A\ 0\ M_0\ M_1 & \longrightarrow & M_0 \\ \mathsf{ifzero}_A\ (n+1)\ M_0\ M_1 & \longrightarrow & M_1 \end{array}$$

$$\begin{array}{rcl} (\lambda x.M)V & \longrightarrow & M[V/x] \\ \mathsf{Y}_{A\to B}V & \longrightarrow & \lambda x^A.V(Y_{A\to B}V)x \end{array}$$

Contextual rules

$$\frac{M \longrightarrow M'}{\mathsf{pred}\, M \longrightarrow \mathsf{pred}\, M'} \qquad \frac{M \longrightarrow M'}{\mathsf{succ}\, M \longrightarrow \mathsf{succ}\, M'} \qquad \frac{M \longrightarrow M'}{\mathsf{ifzero}_A\, M \longrightarrow \mathsf{ifzero}_A\, M'}$$

$$\frac{M \longrightarrow M'}{MN \longrightarrow M'N} \qquad \frac{M \longrightarrow M'}{VM \longrightarrow VM'} \qquad \frac{M \longrightarrow M'}{\mathsf{Y}_{A\to B}\ M \longrightarrow \mathsf{Y}_{A\to B}\ M'}$$