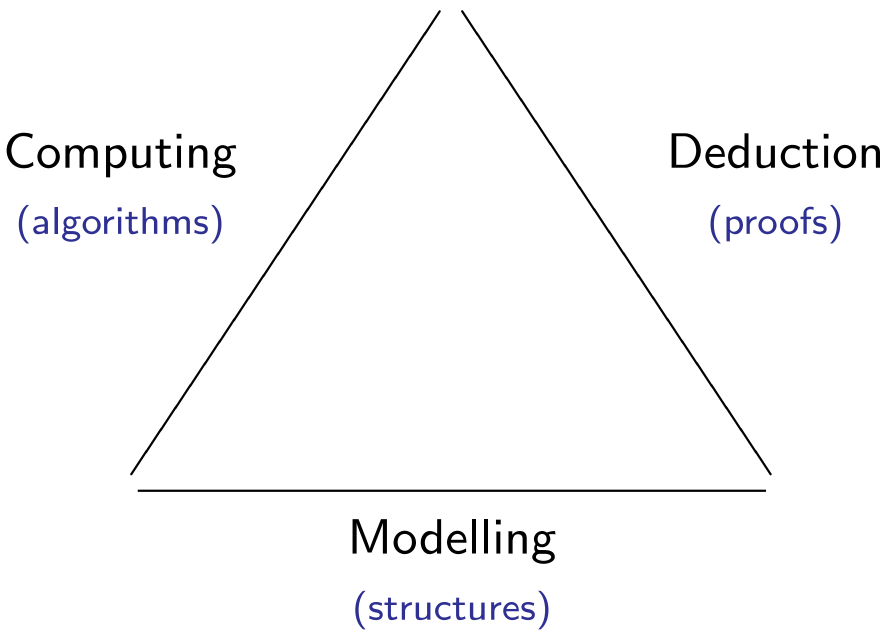
The Impact of the Lambda Calculus

Henk Barendregt

Radboud University
Nijmegen, The Netherlands

Mathematical activity (stylised): modelling, computing, deduction



Solving equations (modern notation) by computing

Egyptian	$ax + b = 0$
Babylonian	$ax^2 + bx + c = 0$

This belongs to *arithmetic*

Also *geometry* was computational: computing areas

Using Pythagoras' theorem to construct right angles

One already knows some *structures*:

numbers:	learned at age 1-3
geometry:	we live in a locally Euclidean space
	our brains are embedded in it
	(Kant's *a priori*)

'There are infinitely many primes' $\forall n \exists p{>}n . p$ is prime

Less emphasis on computing

Euclid could prove:

$$(x+y)^2 = x^2 + 2xy + y^2$$

but not

$$(x+y)^4 = x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4$$

And the Babylonians didn't have deductions

Thesis:	Deduction
Antithesis:	Computing
Synthesis:	Archimede, al-Khôwarizmî combined proving and computing

THEOREM (i) $3.14 < \pi < 3.15$

(ii) $8976 \times 968 = 8688768$

(iii) $\ulcorner n \urcorner \times_a \ulcorner m \urcorner = \ulcorner n \times m \urcorner$

Thesis:	Newton's *Principia* is based on geometry (deduction)
	'It stiffled British mathematics for two centuries'
Antithesis:	Euler, using Leibniz's version of analysis
	made many computational contributions
Synthesis:	Newton partially combined computing and deduction (π in > 30 decimals)
	Augustin-Louis Cauchy (1789-1857)
	made precise 'dealing with arbitrarily small quantities'
	providing an interface between computing and proving

Then mathematics bloomed as never before, leading to applications like

Maxwell's Equations, Relativity Theory and Quantum Physics

Robert Musil (In: *Der Mann ohne Eigenschaften*) about mathematics:

Die Genauigkeit, Kraft und Sicherheit dieses Denkens,
die nirgends im Leben ihresgleichen hat, erfüllte ihn fast mit Schwermut
The precision, force and certainty of this thinking,
unequaled in life, almost filled him with melancholy

Also in the 19-th century the need of structures came up

Solving (using root-expressions)

$ax^3 + bx^2 + cx + d = 0$ (del Ferro, Tartaglia)

$ax^4 + bx^3 + cx^2 + dx + e = 0$ (Descartes)

Polynomials of degree 5 could not be solved similarly (Abel)

Galois made it clear when solutions of polynomials can be expressed as roots

using group theory

Non-Euclidean geometry was invented

....

The wealth of mathematical structures started

to be studied with help of deduction and computation

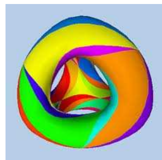
Concepts (structures like groups) became multi-interpretable

Systems for

deduction	Aristotle, Leibniz, Boole, Peirce, Frege [1879] (predicate logic)
modelling	Cantor, Zermelo-Fraenkel [1922] (set theory)
	Whitehead-Russell, de Bruijn, Girard, Coquand-Huet (type theory)
	Eilenberg-MacLane, ..., (category theory)
computing	Thue, Hilbert, Skolem, Herbrand, Kleene,
	Church, Turing [1936] (computability)

Church (1903-1995)
Studying mathematics at
Princeton 1922 or 1924

Supervisor	Oswald Veblen
Suggested topic	find an algorithm for the genus of a manifold $\{\vec{x}\in K^n \mid p(\vec{x}) = 0\}$ (e.g. $K = \mathbb{R}$, $n = 3$)



Church could not do it
Started to wonder what computability is after all
Invented lambda calculus
Formulated Church's Thesis:

Given a function $f : \mathbb{N}^k \to \mathbb{N}$

Then f *is computable iff* f *is lambda definable*

Church tried to use 'functions' to capture both computing and deduction

Lambda calculus terms (λ-terms)

$$\texttt{term} \quad ::= \quad \texttt{var} \mid \texttt{term term} \mid \lambda\texttt{var term}$$

$$\texttt{var} \quad ::= \quad \texttt{x} \mid \texttt{var}'$$

Lambda calculus axiom (computational)

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

For example $(\lambda x.x^2+1)3 = 3^2+1(=10)$.

Lambda calculus axiom (deductional); Γ is a set of terms, $\supset$ is a new constant

$$\frac{\Gamma \vdash (A \supset B) \quad \Gamma \vdash A}{\Gamma \vdash B} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash (A \supset B)}$$

$$\frac{}{\Gamma \vdash M}\text{, if } M \in \Gamma \qquad \frac{\Gamma \vdash A \quad A = B}{\Gamma \vdash B}$$

Proposition. There exist D, B such that

$\mathsf{D}x = xx$ take $\mathsf{D} = \lambda x.xx = \lambda x(xx)$

$\mathsf{B}fgx = f(gx)$ take $\mathsf{B} = \lambda fgx.f(gx) = \lambda f(\lambda g(\lambda x(f(gx))))$

Fixed point theorem. For every F there exists an X such that $FX = X$

Proof. Take $\mathsf{W} = (\mathsf{B}F\mathsf{D})$ and $X = \mathsf{D}\,\mathsf{W}$. Then

$$X = \mathsf{D}\,\mathsf{W} = \mathsf{W}\,\mathsf{W} = \mathsf{B}F\mathsf{D}\,\mathsf{W} = F(\mathsf{D}\,\mathsf{W}) = FX \;\blacksquare$$

Kleene and Rosser showed the system was inconsistent

Curry simplified the proof

Proposition. Any term is provable.

Proof. (Curry's paradox) Given A, let $X = (X \supset A)$, by the fixed point theorem. Now

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{X \vdash X \quad X \vdash X}{X \vdash X \supset A \quad X \vdash X}}{X \vdash A}}{\vdash X \supset A}}{\vdash X}}{\vdash A} \qquad \blacksquare$$

Proposition. Without the deductive part the system is consistent

Lambda terms can express:

$$\left.\begin{array}{l}\bullet\ \text{Computations} \\ \quad\text{- on numbers} \\ \quad\text{- on data types} \\ \bullet\ \text{Infinite processes}\end{array}\right\} \Rightarrow \begin{array}{l}\text{functional programming} \\ \text{languages like ML, Haskell, Clean}\end{array}$$

Reason for its power:

‘meaning/notational-complexity’ arbitrarily high,

with an easy error-correcting interface (typed application)

Church's numerals

$$
\begin{array}{lclcl}
\ulcorner 0 \urcorner & = & \lambda f x . x & & = (\lambda f(\lambda x x)) \\
\ulcorner 1 \urcorner & = & \lambda f x . f x & & = (\lambda f(\lambda x(f x))) \\
\ulcorner 2 \urcorner & = & \lambda f x . f(f x) & & = (\lambda f(\lambda x(f(f x)))) \\
 & & & \ldots & \\
\ulcorner n \urcorner & = & \lambda f x . f^{(n)}(x) & &
\end{array}
$$

There are terms $A_+, A_\times$ satisfying

$$
\begin{array}{lcl}
A_+ \ulcorner n \urcorner \ulcorner m \urcorner & =_\beta & \ulcorner n + m \urcorner \\
A_\times \ulcorner n \urcorner \ulcorner m \urcorner & =_\beta & \ulcorner n \cdot m \urcorner
\end{array}
$$

Take

$$
\begin{array}{lcl}
A_+ & \triangleq & \lambda n m \lambda f x . n f(m f x) \\
A_\times & \triangleq & \lambda n m \lambda f x . m(\lambda y . n f y) x
\end{array}
\quad \text{then} \quad
\begin{array}{lcl}
A_+ n m & = & \lambda f x . n f(m f x) \\
A_\times n m & = & \lambda f x . m(\lambda y . n f y) x
\end{array}
$$

Then

$$
A_+ \ulcorner n \urcorner \ulcorner m \urcorner = \lambda f x . \ulcorner n \urcorner f(\ulcorner m \urcorner f x) = \lambda f x . f^n(f^m x) = \lambda f x . f^{n+m} x = \mathbf{c}_{n+m}
$$

A function $f : \mathbb{N}^k \to \mathbb{N}$ is λ-*definable*

if there exists a lambda term F such that for all $\vec{n} \in \mathbb{N}^k$

$$F \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_k} = \mathbf{c}_{f(\vec{n})}$$

The function p^- predecessor is defined by $p^-(0) = 0,\ p^-(n+1) = n$

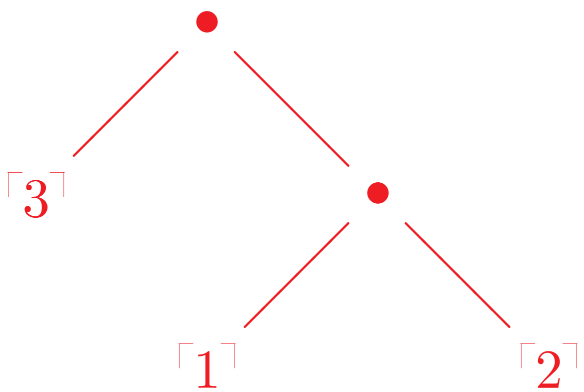
and is λ-defined via

$$P^- \triangleq (\lambda n. n T [\mathbf{c}_0, \mathbf{c}_0] snd),$$

where

$$\begin{array}{rcll}
[M, N] & \triangleq & \lambda z. z M N & \\
fst & \triangleq & \lambda ab.a & [M, N] fst = M \\
snd & \triangleq & \lambda ab.b & [M, N] snd = N \\
T & \triangleq & \lambda p.[S^+(p\, fst), p\, fst] & T[\mathbf{c}_n, \mathbf{c}_m] = [\mathbf{c}_{n+1}, \mathbf{c}_n] \\
 & & & T^n[\mathbf{c}_0, \mathbf{c}_0] = [\mathbf{c}_n, \mathbf{c}_{p^-(n)}] \\
S^+ & \triangleq & \lambda abc.b(abc) & S^+ \mathbf{c}_n = \mathbf{c}_{n+1}
\end{array}$$

A tree like



becomes $\lambda bl.b\,(l\ulcorner 3\urcorner)(b\,(l\ulcorner 1\urcorner)\,(l\ulcorner 2\urcorner))$

A function that mirrors trees is represented as $\texttt{mirror}\ t = \lambda bl.t(\lambda xy.byx)l$:

$$\texttt{mirror}(\lambda bl.b\,(l\ulcorner 3\urcorner)(b\,(l\ulcorner 1\urcorner)\,(l\ulcorner 2\urcorner))) = \lambda b.b\,(b\,(l\ulcorner 2\urcorner)\,(l\ulcorner 1\urcorner))\,(l\ulcorner 3\urcorner)$$

A higher order function 'map' distributing an f over leafs of the tree

$\texttt{map}\ t = \lambda bl.tb(l \circ f)$, where $(l \circ f)\ x = l(fx)$. Then

$$\texttt{map}(\texttt{square})(\lambda b.b\,(b\,(l\ulcorner 2\urcorner)\,(l\ulcorner 1\urcorner))\,(l\ulcorner 3\urcorner)) = \lambda b.b\,(b\,(l\ulcorner 4\urcorner)\,(l\ulcorner 1\urcorner))\,(l\ulcorner 9\urcorner),$$

where $\texttt{square} = \lambda x.A_{\times}xx$

computations	$\rightsquigarrow$	termination
processes	$\rightsquigarrow$	continuation

Simplest continuation

Let $\Delta = \lambda x.xx$. Then

$$\begin{aligned} \Delta\,\Delta &= (\lambda x.xx)\,\Delta \\ &= \Delta\,\Delta \end{aligned}$$

This can be done in more interesting ways

Given $C(\vec{x}, f) = \ldots \vec{x} \ldots f \ldots$, there is a term F such that (general recursion)

$$F\vec{x} = C(\vec{x}, F)$$

Take $F = AA$, with

$$A = \lambda f\vec{x}.C(\vec{x}, f\,f)$$

then

$$AA = \lambda\vec{x}.C(\vec{x}, AA)$$

so

$$F\vec{x} = C(\vec{x}, F)$$

Let $\mathbb{A}$ be a set of symbols. Types over $\mathbb{A}$, notation $\mathbb{T} = \mathbb{T}^{\mathbb{A}}_{\to}$:

$$\boxed{\mathbb{T} \quad = \quad \mathbb{A} \mid \mathbb{T} \to \mathbb{T}}$$

Type assignment

(axiom) $\quad \Gamma \vdash x : A$, if $(x{:}A) \in \Gamma$

$$(\to\mathrm{E}) \quad \frac{\Gamma \vdash M : (A \to B) \quad \Gamma \vdash N : A}{\Gamma \vdash (MN) : B} \qquad (\to\mathrm{I}) \quad \frac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash (\lambda x.M) : (A \to B)}$$

Examples

$$\left.\begin{array}{lcl} \vdash \mathsf{I} & : & (A \to A) \\ \vdash \mathsf{K} & : & (A \to B \to A) \\ \vdash \mathsf{S} & : & (A \to B \to C) \to (A \to B) \to (A \to C) \end{array}\right\} \text{for all } A, B, C \in \mathbb{T}$$

Theorem. $\vdash M : A \;\Rightarrow\; M \in \mathsf{SN}$ (typable terms are strongly normalizing)

Theorem. Type checking is decidable; type reconstruction is computable

Theorem. $\vdash M : A \;\&\; M \twoheadrightarrow N \;\Rightarrow \vdash N : A$ (type checking only at compile time)

I	$\triangleq$	$\lambda x.x$	:	$A \to A$
K	$\triangleq$	$\lambda xy.x$	:	$A \to B \to A$
S	$\triangleq$	$\lambda xyz.xz(yz)$	:	$(A \to B \to C) \to (A \to B) \to A \to C$

From these all closed lambda terms can be defined applicatively

Also with types

Curry: “Hey, these are tautologies” $\mapsto$ Curry-Howard correspondence

Introduction Rules	Elimination Rules
$\dfrac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash (\lambda x{:}A.M) : (A \to B)}$	$\dfrac{\Gamma \vdash F : (A \to B) \quad \Gamma \vdash p{:}A}{\Gamma \vdash (Fp) : B}$
$\dfrac{\Gamma \vdash p : A \quad \Gamma \vdash q : B}{\Gamma \vdash \langle p, q \rangle : (A \,\&\, B)}$	$\dfrac{\Gamma \vdash z : (A \,\&\, B)}{\Gamma \vdash z.1 : A} \quad \dfrac{\Gamma \vdash z : (A \,\&\, B)}{\Gamma \vdash z.2 : B}$
$\dfrac{\Gamma \vdash p : A}{\Gamma \vdash (\mathrm{in}_1\ p) : (A \vee B)} \quad \dfrac{\Gamma \vdash p : B}{\Gamma \vdash (\mathrm{in}_2\ p) : (A \vee B)}$	$\dfrac{\Gamma \vdash p : (A \vee B) \quad \Gamma, x : A \vdash q : C \quad \Gamma, y : B \vdash r : C}{\Gamma \vdash ([\lambda x{:}A.q, \lambda y{:}B.r]p) : C}$
Absurdum Rule	Classical Negation
$\dfrac{\Gamma \vdash p : \bot}{\Gamma \vdash (\mathrm{abs}_A\ p) : A}$	$\dfrac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A}$ here $\neg A {:=} (A \to \bot)$

Intuitionistic/Classical Propositional Logic Natural Deduction Style

Red proofs as λ-terms

For example

$$\vdash \lambda xy.xyy : (A \to A \to B) \to (A \to B)$$

PROPOSITION (Curry-Howard-de Bruijn)

$$\exists M \vdash_{\lambda} {\rightarrow} M : A \iff \vdash_{\mathrm{ML}} A,$$

where $\vdash_{\mathrm{ML}}$ denotes probability in minimal propositional logic

The λ-term M is seen as formalized proof

Can be extended to systems (λ-cube) capturing most mathematics

Proof-checking becomes type-checking (Coq)

Inspired by Ajdukiewicz (and indirectly by Leśniewski)

Curry gave types to syntactic categories

n noun/subject s sentence			
$n \to n$	‘red hat’ (adjective)	$(n \to s) \to (n \to s)$	adverbs
$(n \to n) \to n$	‘redness’	$(n \to s) \to s$	quantifiers
$n \to (n \to n)$	‘(John and Henry) are brothers’		
$n \to s$	‘Mary sleeps’		
$n \to n \to s$	‘Mary kisses John’		
$s \to s$	‘not(Mary kisses John)’		
More complex cases			
$(n \to n) \to (n \to n) \to (n \to n)$		‘slightly large’	
$((n \to n) \to (n \to n)) \to (n \to n) \to (n \to n)$		‘slightly too large’	

Do not know whether Montague had seen this

1900-2000

The computing and deduction traditions again diverged:

Computer Algebra systems versus Proof Checking systems

2000-2100

Certified Mathematics/Computer Science, will unify the two

Interface computation—deduction: two styles

$\dfrac{\Gamma \vdash p : A(t)}{\Gamma \vdash p : A(s)}\,,\ t = s$	$\dfrac{\Gamma \vdash p : A(t) \quad \Gamma \vdash C : t = s}{\Gamma \vdash f(p, C) : A(s)}$
Poincaré Principle (Coq)	Ephemeral proof-objects (HOL)
$2 + 2 = 4$ doesn't need a proof	looong proofs are checked but not stored

Computing	Proving	Modeling
λ−term	λ−term	type

$$\Gamma \vdash M \; : \; A$$

$$\vdash \lambda\Gamma.M \; : \; \Pi\Gamma.A$$

Certifications

Fundamental Theorem of Algebra	Geuvers, Wiedijk, Zwanenburg, Pollack and Niqui	Coq
Fundamental Theorem of Calculus	Cruz-Filipe	Coq
Correctness Buchberger's algorithm	Person, Théry	Coq
Primality of $902625808338499686044936607214230780196 3$	Oostdijk, Caprotti	Coq
Correctness of Fast Fourier Transform	Capretta	Coq
Book "Continuous lattices" (in part)	Bancerek et al.	Mizar
Impossibility of trisecting angles	Harrison	HOL-light
$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$	Harrison	HOL-light
Prime Number Theorem	Avigad, Harrison	Isabelle-HOL
Four Colour Theorem	Gonthier	Coq
Jordan Curve Theorem	Hales	HOL
Primality of >100 digit numbers	Grégoire, Théry, Werner	Coq
$\lambda\beta\boldsymbol{\eta}$SP conservative over $\lambda\beta\boldsymbol{\eta}$	Stoevring	Twelve
Proof of the Kepler conjecture	Hales [2014]	HOL-light
ARM6 processor	Fox [2003]	HOL
seL4 Operating system	Klein et al [2009]	
C-compiler	Leroy et al [2009]	Coq

Cambridge University Press, June 2013

Summary in ≤ 20 words of 698 pages

This handbook with exercises reveals in formalisms

hitherto mainly used for designing and verifying software and hardware

unexpected mathematical beauty

1. Let g, h be λ-definable functions Define

$$\begin{aligned} f(0,\vec{x}) &= g(\vec{x}) \\ f(n+1,\vec{x}) &= h(n,\vec{x},f(n,\vec{x})) \end{aligned}$$

Show that f is λ-definable. [Hint. First find a term P such that

$$P\mathbf{c}_n\mathbf{c}_{\vec{m}} = [\mathbf{c}_{f(n,\vec{m})}, \mathbf{c}_n].]$$

2. Let g be λ-definable, such that $\forall\vec{m}\exists n.g(n,\vec{m}) = 0$. Define

$$f(\vec{m}) = \mu x.g(x,\vec{m}) = 0.$$

Show that f is λ-definable. [Hint. Use the fixed point theorem.]

3. Sketch a proof that all λ-definable functions are computable.

4. Conclude that the computable and λ-definable functions form the same class.