Lambda-Calculus : from logic to computation

Wendy Hammache

IRPhiL, Université Jean Moulin Lyon 3

PROGRAMme Autumn workshop: formalisms at the interface with machines, languages and systems, Bertinoro
October 17, 2018

Lambda-calculus

Nowadays we say about lambda-calculus :

It is the first programming language
It is the formalism underlying every functional programming language
It embodies the formalism to write every program we can code with traditional programming languages

Lambda-calculus

Why would we want to create a programming language ?
as computers did not exist yet

What could be the resaons to create a language such as a programming one ?
Motivations ? Usage ?

Outline

- Context : developments of the notion of function
- Age 1 : Lambda-formalism in a system for foundational purposes
- Age 2 : Constitution of lambda-calculus
- Age 3 : Introduction of types

Context before the birth of lambda-calculus

1. Frege and Schönfinkel contributed to develop the notion of function in several directions

Extension	Allow functions to take functions as arguments (Frege, Schönfinkel)
Currying	Introduction of a device to represent function of several arguments as functions of one argument ([Frege, 1891], [Frege, 1893] [Schönfinkel, 1924])
Liberalization	Allow functions to be applied to themselves
Reduction	Reduction of most of notions to that of function (eg. notion of class) Reduction of the number of symbols
Purification	Get rid of free variables
Abstraction	Fregean course-of-value (1891, 1893) : ancester of Churchean λ-abstraction *Ex :* *Church lambda-abstraction* $\lambda x(x+2)$ *Frege course-of-value* $\alpha'(\alpha+2)$

Context before the birth of lambda-calculus

2. Schönfinkel (1924) builds a "calculus of functions"

> *It will [. . .] be necessary to [. . .] develop first a kind of function calculus [Funktionenkalkül]; we are using this term here in a sense more general than is otherwise customary.*
> *[Schönfinkel, 1924, p.359]*

and tries to express a large set of functions
But λ-calculus as combinatory logic were not born as function calculi as such but as parts of more general formal systems

Church's functional system : Set of postulates (1932-33)

Motivations

- Foundational perspective :
  - avoid contradictions
  - be adequate to build number theory
- Surpass old solutions (even theory of types)
- Give a system that can prove its own consistency

> *the method of Russell for avoiding the familiar paradoxes of mathematical logic, or that of Zermelo, [. . .] appear somewhat artificial*
> *[Church, 1932, p.347]*

> *Dr. von Neumann called my attention last fall to your paper entitled "Über formal unentscheidbase Sätze der Principia Mathematica". I have ben unable to see, however, that your conclusions in §4 of this paper apply to my system. Possibly your argument can be modified so as to make it apply to my system, but I have not been able to find such a modification of your argument.*
> *Unpublished letter from Church to Gödel - July 27, 1932*

Church's functional system : Set of postulates (1932-33)

Features

1. Formal system (*à la* Hilbert) axiomatized
2. Notion of function taken as primitive
   - Rules of syntax define functional devices
   - Logical constants (connectives) are treated as functions : $A \wedge B$ is (&(A))B
3. The system includes some features of what we know nowadays as λ-calculus
4. Notion of formula and wff are quite liberal
5. 37 and, after amendments, 32 postulates (axioms of the system)
6. 5 rules of procedure (rules of inference)

Church's functional system : Set of postulates (1932-33)

5 Rules of procedure

1. α-conversion (simple rule of substitution)
2. β-reduction : *rule of conversion II* [Church, 1932] ; *R-conversion* [Rosser, 1935] ; *reduction* [Church and Rosser, 1936]

> *II. If J is true, if M and N are well-formed, if the variable x occurs in M, and if the bound variables in M are distinct both from the variable x and from the free variables in N, then K, the result of substituting* $S^x_N M|$ *for a particular occurrence of* $\{\lambda x.M\}(N)$ *in J, is also true.*
> *[Church, 1932, p.355]*

3. β-expansion (reverse of β-reduction)
4. A rule to state existence. In contemporary notation $F(A) \vdash \exists x F(x)$
5. Rule of deduction (in contemporary notation) : $F \to G, F(A) \vdash G(A)$

Remark

All these rules are first given in terms of truth : rather than $A \vdash B$ we say if A is true, then B is true - even for conversion rules.
The symbol $\vdash$ for its derivation denotation was introduced by [Rosser, 1935]

Church's functional system : Set of postulates (1932-33)

Foundational problems solved (or not)

The system of Church aimed to :

- avoid paradoxes without the notion of type, by the introduction a restriction of the excluded-middle

> *We introduce a certain restriction on the law of excluded middle as a means of avoiding the paradoxes connected with the mathematics of the transfinite.*
> *[Church, 1932, p.346]*

- get rid of the Russell paradox by giving it a new formulation
- get rid of the result of Gödel (failed)

> *Correspondence [with Gödel] proves mostly that I was slow in comprehension - but I got there in the end.*
> *Undated note on the letter from Church to Gödel - July 27, 1932*

What about combinatory logic ?

The first version developped by Curry :

- Was also an axiomatized system close to Church's one
- Included combinations as Schönfinkel's ones

  Schönfinkel's combinators C, S, Z, T, I
  Curry's combinators K, S, B, C, I
- Was equivalent to Church's system (result of [Rosser, 1935]).

Kleene-Rosser Paradox

[Kleene and Rosser, 1935]
Shows that in Church's and Curry's systems are inconsistent
"every well-formed formula with no free variables is provable"
[Kleene and Rosser, 1935, p.635, Theorem C]

- Based on Richard's paradox
- Impossible to get rid of this contradiction as easily as for Richard's one
- Building of a function G providing a *translation* of some intuitive notions to formal ones (cf. Gödel numbering method).
- Possible because of the great liberalism of these systems

> *In order that a G exist, the logic must permit great freedom in the expression of its formulas as values of functions. Logics have been proposed which have this freedom, e.g. a combinatory logic of Curry and a system of Church. Both of these systems are inconsistent.*
> *[Kleene and Rosser, 1935, p.631]*

Doubts : is the project of such a system feasible ?

Church to Curry, January 19, 1935 (unpublished letter)

Is there a system of symbolic logic which embodies your fundamental combinatory rules, is free from contradiction, and is at the same time adequate for a substantial portion of mathematics ?

Curry to Church, April 9, 1935 (unpublished letter)

I don't know. [. . .] Of course I do not have any such theory readily at hand. [. . .] But I think it is equally fruitless to speculate as to the answer to your question[. . .] Let us see what we can find out. Suffice it to say, then, that the possibility is by no means excluded by any result in the field of mathematical logic[. . .]. Rosser's result, however, marks something of a turning point in the development of the theory ; he has indeed shown that certain hypotheses, which I had entertained, are untenable, and that certain alernatives, in the development of combinatory logic, will have to be excluded. This may require a considerable modification in my present set-up before we are through : but that is a help, not an hindrance, to the development of program, since what we are all after is the discovery of truth.

Consequence of the Kleene-Rosser paradox

Impossibility to combine two forms of completeness in a system

- deductive completeness
- combinatorial completeness

It is then required to investigate

- Either foundational problems
- Either computational perspectives

Probably due to the context.

Consequence of the Kleene-Rosser paradox

Impossibility to combine two forms of completeness in a system

- deductive completeness
- combinatorial completeness

It is then required to investigate

- Either foundational problems
- Either computational perspectives

Probably due to the context.

Consequence of the Kleene-Rosser paradox

Impossibility to combine two forms of completeness in a system

- deductive completeness
- combinatorial completeness

It is then required to investigate

- Either foundational problems
- **Either computational perspectives**

Probably due to the context.

What lambda-calculus is

Remaining part when leaving off most of features.

> *I had a scheme that had the lambda-calculus as part of it. After publishing a couple of attempts that actually lead to inconsistency, I decided that it couldn't be put through, so the lambda-calculus is all that is left of that.*
> *Oral history interview with Alonzo Church by W. Aspray, 1984*

A calculus of functions

Idem for combinatory logic.

That is, we keep only :

- λ-notation
- 3 of the 5 rules of procedure
- some other devices developped then to make other combinations

Aspects of computation already explored

Researches on Church's system permitted to :

1. Formalize a rule of substition that can be regarded as computational process (by reduction) in a simple way
2. Give an empty shell for further investigations

   The initial set of postulates must of themselves define the system as a formal structure [. . .]. There may, indeed, be other applications of the system than its use as a logic.
   [Church, 1932, p.349]

3. Introduce a definition for positive integers and usual basic operations [Church, 1933]
4. Investigate what could be a "computationally exhaustive" system

Difficulties about the K-function

4. Investigate what could be a "computationally exhaustive" system

Church's first system requirement

In λxM, x must occur (free) in M

Some functions cannot be defined

K-function (strong form) - taken by Curry from Schönfinkel

$KXY = X$ whatever X and Y are

- K-function = enables weak-normalization.
- In 30's terminology : it complicates a proof of freedom from contradiction.

K-function (weak form) - accepted by Church

$KXY = X$ iff Y has a normal form.

More developments of the lambda-calculus as such

After the extraction of λ-calculus as such, some new developments were made :

- Attempts to characterize all functions which have a normal form (without the use of types)

> *[. . .] the type condition is a sufficient condition, but far from a necessary condition, that a formula have a normal form. I regard it as obvious that a formula has an inutitive meaning if and only if it has a normal form (i.e. at least in the "pure" lambda-formalism whose only proper symbols are variables). It is strongly suggested to liberalize by replacing the type condition by the less stringent condition of having a normal form. Unfortunately, the latter condtion is non-finitary. One might try to find a finitary equivalent of the normal form condtion, not in the sense of an effective criterion for having a normal form (which is impossible), but in the sense that in a particular system it is made to work out so that provable formulas do in fact always always have a normal form.*
>
> *Unpublished letter from Church to Quine, June 3, 1938*

- Rise of the idea of "computational consistency"

Church's simple type theory (1940)

What about foundational preoccupations ?

Church's simple type theory is an incorporation of the λ-formalism, in the simple type theory [Church, 1940]
This is not typed λ-calculus

> *The purpose [. . .] is to give a formulation of the simple theory of types which incorporates certain features of the calculus of λ-conversion*
> *[Church, 1940, p.56]*

"Partial incorporation (...) of the calculus of λ-conversion into the theory of types"

Motivations : why a lambda theory of types ?

- Foundational perspective
- Simplication of early works on types
- Introduction of the λ-formalism in the 'great history of theory of types'
- Build a system of *Principia Mathematica*, in terms of λ-notation

> *I have finally completed the task of writing up for publication my version of the system of Principia Mathematica (with simple theory of types and incorporating lambda-conversion).*
> *Unpublished letter from Church to Turing - March 22, 1940*

- Give a fregean-like formulation of the theory of types, with a central notion of function

> *In the paper now under review [Church, 1940], finally, Church adapts the theory of types for the first time to the Fregean approach.*
> *[Quine, 1940]*

Direct influences Frege, Russell, Carnap & Chiwstek

Other works around types since Russell Carnap (syntaxical categories), Gentzen (Stufenlogik), Turing (applied type theory), Curry (notion of Functionality)

Definitions of Church's type theory

Definition of types

ι is the type of the individuals
o is the type of the propositions
If α and β are types, $(\alpha\beta)$ is a type

Notation of Church's types of functions

$F_{\beta\alpha}$
Note :
Type of the argument is α
Type of the result is β

Features of Church's type theory

- Explicit representation of types that differs from Russell's one
- Set-theoretic point of view

  Each type is a domain of things that are called the members of the type.
  Church, (unpublished) Frist draft paper developping a formalized language, June 1985 (N023)

- Empty shell (abstract formal theory)

  We purposely refrain from making more definite the nature of the types o and ι, the formal theory admitting of a variety of interpretations in this regard[...], and indeed other and quite different interpretations are possible (formal consistency assumed).
  [Church, 1940, p.57]

- A step to data types : boolean values

  The members of the type o are the two truth values, truth and falsehood.
  Church, (unpublished) Frist draft paper developping a formalized language, June 1985 (N023)

- A step to polymorphism : $N_{o\alpha' x_{\alpha'}} \subset N_{o\alpha'' (T_{\alpha'' \alpha' x_{\alpha'}})}$ ([Church, 1940], Theorem 31^{α})

Unification : lambda calculus + types

Lambda-calculus + features of type theory = typed lambda-calculus
Note that : Church has never published a theory of typed lambda-calculus

Features and usage

- Rules for typing : introduced by Church1951

(1) a formula consisting of a single proper symbol is well-formed and has the type indicated by the subscript ;
(2) if x_β is a variable of type β and M_α is a well-formed formula of type α, then $(\lambda x_\beta M_\alpha)$ is a well-formed formula having the type $\alpha\beta$;
(3) if $F_{\alpha\beta}$ and A_β of types $\alpha\beta$ and β respectively, then $(F_{\alpha\beta} A_\beta)$ is a well-formed formula having the type α.
[Church, 1951, p.8-9]

- Church's usage in linguisitc theory
- Other investigations in theory of computation (Scott)

Conclusion

- λ-calculus as we know it nowadays was never invented as such
- Kleene-Rosser paradox points the begining and not the end of λ-calculus
- History of λ-calculus shows a transition from logico-foundational preoccupations to computational ones.
- Elasticity and universality of λ-calculus results from the abstract character of λ-fromalism and of systems in which it was stated


Church, A. (1932).
A set of postulates for the foundation of logic.
Annals of Mathematics, 33(2) :346–366.


Church, A. (1933).
A set of postulates for the foundation of logic (2nd paper).
Annals of Mathematics, 34(4) :839–864.


Church, A. (1940).
A formulation of the simple theory of types.
The Journal of Symbolic Logic, 5(2) :56–68.


Church, A. (1951).
A Formulation of the Logic of Sense and Denotation, volume 17, pages 3–25.
New York, Liberal Art Press.


Church, A. and Rosser, J. B. (1936).
Some properties of conversion.
Transactions of the American Mathematical Society, 39(3) :472–482.


Frege, G. (1891).
Fonction et concept.
Seuil.


Frege, G. (1893).

Grundgesetze der Arithmetik, volume I.
Verlag Hermann Pohle, Jena.

Kleene, S. C. and Rosser, J. B. (1935).
The inconsistency of certain formal logics.
Annals of Mathematics, 36(3) :630–636.

Quine, W. V. (1940).
review : A. church, a formulation of the simple theory of types.
J. Symbolic Logic, 5(iss.3) :114–115.

Rosser, J. B. (1935).
A mathematical logic without variables. i.
Annals of Mathematics, 36(1) :127–150.

Schönfinkel, M. (1924).
On the building blocks of mathematical logic.