The Curry-Howard correspondence journey

Silvia Ghilezan

University of Novi Sad
Mathematical Institute SASA, Serbia

Lecture 1

Proof Society International School
Birmingham, September 2024

correspondence

logic **computation**

proofs	–	terms
formulae	–	types
rules	–	reductions

Curry - Howard correspondence

INTUITIONISTIC LOGIC	COMPUTATION
Axiomatic system	Combinatory Logic
Natural Deduction	λ-calculus
Sequent system	sequent-like λ-calculi

coincidence

- 1950s Curry
- 1968 (1980) Howard - formulae-as-types
- 1970s Lambek - CCC Cartesian Closed Categories
- 1970s de Bruijn - AUTOMATH
- 1970s Martin-Löf - Type Theory

Journey

LOGIC	**COMPUTATION**
intuitionistic	λ-calculus combinatory logic
second-order	polymorphism
predicate	λ cube
classical	$\lambda\mu$-calculi
???	intersection types
communication	process calculi
??? federated (machine) learning	???

Roadmap Lecture 1

Logical systems

Lambda Calculus and Combinatory Logic

Simple (functional) types

Correspondence: Intuitionistic Logic and Computation

Roadmap

Logical systems

Lambda Calculus and Combinatory Logic

Simple (functional) types

Correspondence: Intuitionistic Logic and Computation

Logical systems

LOGIC	
Axiomatic system	Hilbert style
Natural Deduction	Gentzen, Prawitz
Sequent system	Gentzen

Intuitionistic logic - axiomatic system

Brouwer (early 1900), Heyting (1960s)

$$\sigma \ ::= \ \alpha \mid (\sigma \to \sigma)$$

α ranges over a countable set of propositional variables

- **Axioms**

  (Ax 1) $\sigma \to (\tau \to \sigma)$

  (Ax 2) $(\sigma \to (\tau \to \rho)) \to ((\sigma \to \tau) \to (\sigma \to \rho))$

- **Inference rule**

  (*MP*)
  $$\frac{\sigma \to \tau \quad \sigma}{\tau}$$

- $\neg, \ \wedge, \ \vee, \ \leftrightarrow$ connectives are independent

- **BHK interpretation** - Brouwer, Heyting, Kolmogorov interpretation of logical connectives
  “math is a cognitive construct rather than a type of objective truth"
- Provable, theorems
  - $\vdash \sigma \to \sigma$, identity
  - $\vdash \sigma \to \neg\neg\sigma$, double negation introduction
  - $\vdash \neg\sigma \wedge \neg\tau \leftrightarrow \neg(\sigma \vee \tau)$, de Morgan law $(\frac{2}{4})$
  - $\vdash \neg\sigma \vee \neg\tau \to \neg(\sigma \wedge \tau)$, de Morgan law $(\frac{1}{4})$
- Not provable
  - $\nvdash ((\sigma \to \tau) \to \sigma) \to \sigma$, Peirce law
  - $\nvdash \sigma \vee \neg\sigma$, law of excluded middle (tertium non datur)
  - $\nvdash \neg\neg\sigma \to \sigma$, double negation elimination
  - $\nvdash \neg(\sigma \wedge \tau) \to \neg\sigma \vee \neg\tau)$, de Morgan law $(\frac{1}{4})$

Classical logic - axiomatic system

- **Axioms**

  (Ax 1) $\sigma \to (\tau \to \sigma)$

  (Ax 2) $(\sigma \to (\tau \to \rho)) \to ((\sigma \to \tau) \to (\sigma \to \rho))$

  (Ax 3) $(\neg\sigma \to \neg\tau) \to ((\neg\sigma \to \tau) \to \sigma)$

- **Inference rule**

  (*MP*) $$\frac{\sigma \to \tau \quad \sigma}{\tau}$$

- $\neg,\ \wedge,\ \vee,\ \leftrightarrow$ connectives are dependent
- Interdefinability of connectives: minimal bases for defining all connectives:
  - $\{\neg, \to\}$, $\{\neg, \wedge\}$, $\{\neg, \vee\}$
  - $\{\downarrow\}$ Peirce arrow (NOR)
  - $\{\uparrow\}$ Sheffer stroke (NAND)

Roadmap

Logical systems

Lambda Calculus and Combinatory Logic

Simple (functional) types

Correspondence: Intuitionistic Logic and Computation

Models of computation

- Alonzo Church:
  - theory of functions - formalisation of mathematics (inconsistent)
  - successful model for computable functions - **λ-calculus**
  - **simply typed λ-calculus**
- Haskell Curry:
  - elimination of variables in logic - Moses Schönfinkel (1921)
  - successful model for computable functions - **Combinatory logic**
  - **Combinatory logic with types**
- Alan Turing :
  - formalisation of the concepts of algorithm and computation
  - **Turing Machines**

λ-calculus - expressiveness

- Expressiveness - Effective computability (mid 1930s)
  - **(Curry)** Equivalence of λ-calculus and Combinatory Logic
  - **(Kleene)** Equivalence of λ-calculus and recursive functions
  - **(Turing)** Equivalence of λ-calculus and Turing machines

Combinatory logic

Curry, Schönfinkel (1921)

Syntax

$$M \ ::= \ x \mid \mathbf{S} \mid \mathbf{K} \mid \mathbf{I} \mid MM$$

Reduction rules

$$\begin{array}{lcl} \mathbf{I}M & \longrightarrow & M \\ \mathbf{K}MN & \longrightarrow & M \\ \mathbf{S}MNP & \longrightarrow & MP(NP) \end{array}$$

Example $\mathbf{SKK} = \mathbf{I}$

No bound variables ✓

λ-calculus - theory of functions

Syntax

$$M ::= x \mid c \mid (MM) \mid (\lambda x.M)$$

Reduction rules

α-reduction (renaming):

$$\lambda x.M \longrightarrow_\alpha \lambda y.M[x := y], \quad y \notin FV(M)$$

β-reduction (function evaluation):

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

Example

$(\lambda x.x)5 \rightarrow 5$ identity function

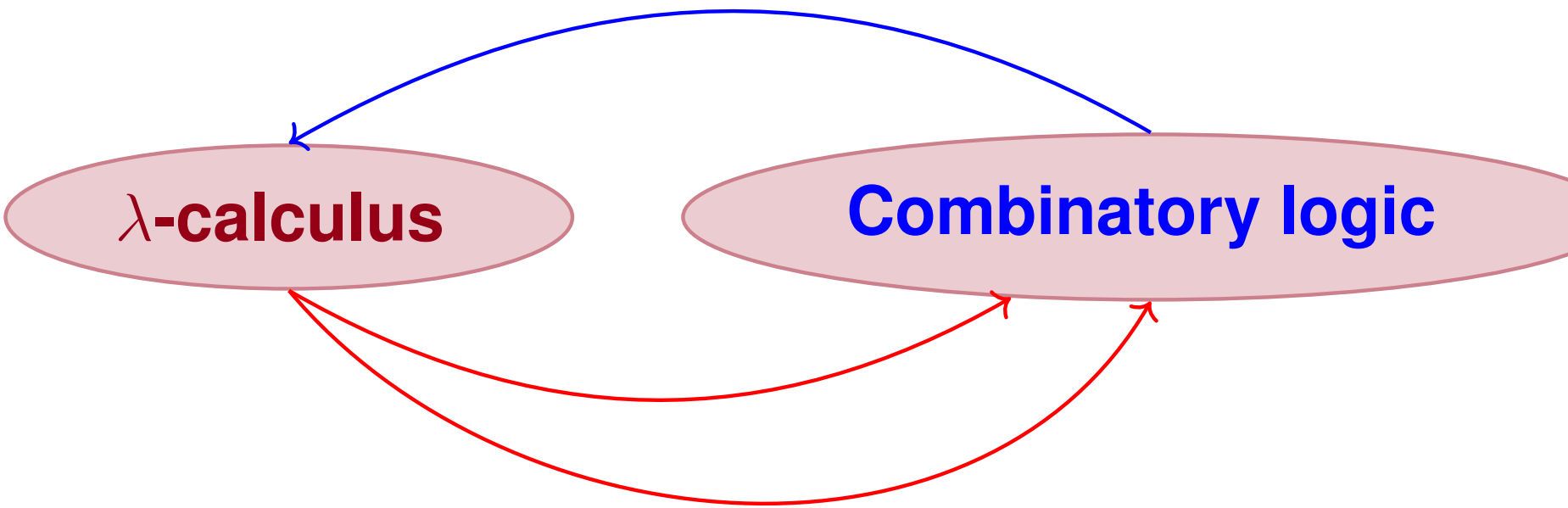
$(\lambda x.x^2 + 1)3 \rightarrow 3^2 + 1 = 10$

Equivalence of expressiveness

$$\mathbf{I} \equiv \lambda x.x$$

$$\mathbf{K} \equiv \lambda xy.x$$

$$\mathbf{S} \equiv \lambda xyz.xz(yz)$$



Running example

$xyzx$

$\lambda x.zx$

$\mathbf{I} \equiv \lambda x.x$	combinator **I**
$\mathbf{K} \equiv \lambda xy.x$	combinator **K**
$\mathbf{S} \equiv \lambda xyz.xz(yz)$	combinator **S**
$\Delta \equiv \lambda x.xx$	selfapplication
$\mathbf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$	fixed point combinator
$\Omega \equiv \Delta\Delta \equiv (\lambda x.xx)(\lambda x.xx)$	higher-order function

Properties

confluence	✓
normal forms	uniqueness (if exists)
normalisation	✗
strong normalisation	✗
fixed point theorem	✓
expressiveness	Turing machines, recursive functions

Properties - Confluence

Theorem (Church-Rosser theorem)

If $M \longrightarrow\!\!\!\!\!\rightarrow N$ and $M \longrightarrow\!\!\!\!\!\rightarrow P$, then there exists S such that $N \longrightarrow\!\!\!\!\!\rightarrow S$ and $P \longrightarrow\!\!\!\!\!\rightarrow S$

The proof is deep and involved.

Corollary

- *If $M \longrightarrow\!\!\!\!\!\rightarrow N$ and $M \longrightarrow\!\!\!\!\!\rightarrow P$, then $N = P$*
- *The order of the applied reductions is arbitrary and always leads to the same result*
- *Reductions can be executed in parallel (parallel computing)*

Proof.

Normal forms

- $N \in \Lambda$ is a normal form (NF) if there is no S such that $N \longrightarrow S$
- $P \in \Lambda$ is normalising (has a normal form) if $P \longrightarrow\!\!\!\!\!\rightarrow N$ and N is a normal form, then N is a NF of P
- $P \in \Lambda$ is strongly normalising (SN) if all reductions of P are finite

Notation: $\longrightarrow\!\!\!\!\!\rightarrow$ will denote $\longrightarrow\!\!\!\!\!\rightarrow_\beta \cup \longrightarrow_\alpha$

Theorem (uniqueness of NF)

Every lambda term has at most one normal form

Proof.

Exercise

$\mathbf{I} \equiv \lambda x.x$	normal form NF
$\mathbf{K} \equiv \lambda xy.x$	normal form NF
$\mathbf{S} \equiv \lambda xyz.xz(yz)$	normal form NF
$\Delta \equiv \lambda x.xx$	normal form NF
$\mathbf{KI}(\mathbf{KII})$	strongly normalizing SN
$\Omega \equiv \Delta\Delta \equiv (\lambda x.xx)(\lambda x.xx)$	unsolvable
$\mathbf{KI}\Omega$	normalizing N
$\mathbf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$	head normalizing HN (solvable)

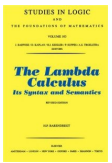
$\mathbf{KI}(\mathbf{KII}) \rightarrow \mathbf{KII} \rightarrow \mathbf{I}$

$\mathbf{KI}(\mathbf{KII}) \rightarrow \mathbf{I}$

$\Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \ldots$

$\mathbf{KI}\Omega \rightarrow \mathbf{I}$

References



H.P. Barendregt.

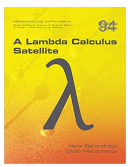
Lambda Calculus: Its syntax and Semantics.

North Holland, 1984.

F. Cardone, J. R. Hindley

History of Lambda-calculus and Combinatory Logic

Handbook of the History of Logic. Volume 5. Logic from Russell to Church Elsevier, 2009, pp. 723-817 (*online 2006*)



H.P. Barendregt, G. Manzonetto

A Lambda Calculus Satellite

College Publications, 2022.

Roadmap

Logical systems

Lambda Calculus and Combinatory Logic

Simple (functional) types

Correspondence: Intuitionistic Logic and Computation

Simple (functional) types

Motivation

- “Disadvantages" of the untyped calculus:
  - infinite computation - there exist terms without a normal form
  - meaningless applications - it is allowed to create terms like **sin log**
- Types are syntactical objects that can be assigned to terms
  - Reasoning with types was present in the early work of Church and Curry on untyped calculi
- two typing paradigms:
  - *à la* Curry - implicit type assignment (*lambda calculus with types*)
  - *à la* Church - explicit type assignment (*typed lambda calculus*)

Simple (functional) types

- Language of types **Type**

$$\sigma \ ::= \ a \mid \sigma \to \sigma$$

- Type assignment

$M \in \Lambda(CL)$ $\qquad\qquad$ $\sigma \in$ **Type**

$$M : \sigma$$

$\lambda \rightarrow$ - the type system

à la Curry and à la Church

- Axiom

$$(Ax) \qquad \frac{}{\Gamma, x : \sigma \vdash x : \sigma}$$

- Rules

$$(\rightarrow_{elim}) \qquad \frac{\Gamma \vdash M : \sigma \rightarrow \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \rightarrow \tau}\,(\rightarrow_{intr}) \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma.M : \sigma \rightarrow \tau}$$

à la Curry à la Church

Running example: types

M	Type
$\mathbf{I} \equiv \lambda x.x$	$\sigma \to \sigma$
$\mathbf{K} \equiv \lambda xy.x$	$\sigma \to (\tau \to \sigma)$
$\mathbf{S} \equiv \lambda xyz.xz(yz)$	$(\sigma \to (\tau \to \rho)) \to ((\sigma \to \tau) \to (\sigma \to \rho))$
$\Delta \equiv \lambda x.xx$	NO
$\mathbf{Y}$	NO
$\Omega \equiv \Delta\Delta$	NO

Properties

confluence	✓
normal forms	✓
normalisation	✓
strong normalisation	✓
fixed point theorem	✗
expressiveness	extended polynomials

Combinatory logic with simple types

- **Axioms**
  - **(Ax 1)** $\mathbf{K} : \sigma \to (\tau \to \sigma)$
  - **(Ax 2)** $\mathbf{S} : (\sigma \to (\tau \to \rho)) \to ((\sigma \to \tau) \to (\sigma \to \rho))$
- **Inference rule**

$$(MP) \qquad \frac{M : \sigma \to \tau \quad N : \sigma}{MN : \tau}$$

- **Y** and Ω are not typable

3 Type?

Type checking: given M and σ

$$(M : \sigma)?$$

Type inference (typability, type synthesis): given M

$$M :?$$

Type inhabitation (term, program synthesis) : given σ

$$? : \sigma$$

$\lambda \rightarrow$ 3 Type?

Theorem

In $\lambda \rightarrow$

- *Type checking* $((M : \sigma)?)$ *is decidable*
- *Type inference* $(M :?)$ *is decidable*
- *Type inhabitation* $(? : \sigma)$ *is decidable*

Roadmap

Logical systems

Lambda Calculus and Combinatory Logic

Simple (functional) types

Correspondence: Intuitionistic Logic and Computation

Axioms and Combinators

Intuitionistic logic ✓

Classical logic ✗

- **Axioms**

(Ax 1) **K** $:\sigma \to (\tau \to \sigma)$

(Ax 2) **S** $:(\sigma \to (\tau \to \rho)) \to ((\sigma \to \tau) \to (\sigma \to \rho))$

(Ax 3) ✗ $: (\neg\sigma \to \neg\tau) \to ((\neg\sigma \to \tau) \to \sigma)\ (\neg\sigma \to \neg\tau) \to$

- **Inference rule**

(MP) $$\frac{M :\sigma \to \tau \quad N :\sigma}{MN :\tau}$$

Theorem (Curry Howard correspondence)

$\vdash \sigma(IL)$ *if and only if* $\vdash M : \sigma$

Theorem (Curry Howard correspondence)

$$\vdash \sigma \ (\mathit{PROP}) \quad \textit{if and only if} \quad \vdash M : \sigma$$

The correspondence

- holds for intuitionistic logic
- does not hold for classical logic

Natural deduction and λ

Intuitionistic logic (minimal) - Natural deduction, Gentzen 1930s

- Axiom

$$(Ax) \qquad \frac{}{\Gamma, \sigma \vdash \sigma}$$

- Rules

$$(\rightarrow_{elim}) \qquad \frac{\Gamma \vdash \sigma \rightarrow \tau \qquad \Gamma \vdash \sigma}{\Gamma \vdash \tau}$$

$$(\rightarrow_{intr}) \qquad \frac{\Gamma, \sigma \vdash \tau}{\Gamma \vdash \sigma \rightarrow \tau}$$

$\lambda \rightarrow$ and logic

Intuitionistic logic (minimal) - Natural deduction, Gentzen 1930s

- Axiom

$$(Ax) \qquad \frac{}{\Gamma, x : \sigma \vdash x : \sigma}$$

- Rules

$$(\rightarrow_{elim}) \qquad \frac{\Gamma \vdash M : \sigma \rightarrow \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

$$(\rightarrow_{intr}) \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \rightarrow \tau}$$

Curry-Howard correspondence

Intuitionistic logic vs computation

$$\vdash \sigma \quad \Leftrightarrow \quad \vdash M : \sigma$$

Theorem

A formula is provable in minimal intuitionistic logic
if and only if it is inhabited in $\lambda \rightarrow$

formulae	–as–	types
proofs	– as –	terms
proofs	–as–	programs
proof normalisation	–as–	term reduction
cut elimination	–as–	term reduction

- ✓ BHK - Brouwer, Heyting, Kolmogorov interpretation of logical connectives is formalized by the Curry-Howard correspondence
- $\mathcal{P}(A \to B)$ are the maps from $\mathcal{P}(A)$ into $\mathcal{P}(B)$
- Pierce's law not inhabited ?

Curry - Howard correspondence

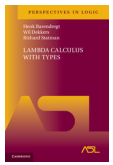
INTUITIONISTIC LOGIC	COMPUTATION
Axiomatic system	Combinatory Logic
Natural Deduction	λ-calculus
Sequent system	sequent-like λ-calculi

References

H.P. Barendregt

Lambda calculi with types

Handbook of Logic in Computer Science, Oxford University Press, 1993



H.P. Barendregt, W. Dekkers, R. Statman

Lambda Calculus with Types

Cambridge University Press 2013

M. H. B. Sörensen, P. Urzyczyn

Lectures on the Curry-Howard isomorphism

MIT Press 2002

H. Barendregt, S. Ghilezan

Lambda terms for natural deduction, sequent calculus and cut elimination

Journal of Functional Programming 10: 121-134 (2000)