Proofs-as-programs: from logic to AI

Silvia Ghilezan

University of Novi Sad
Mathematical Institute SASA, Serbia

LAP 2024
IUC, Dubrovnik
23-27 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

- 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	???

Curry - Howard correspondence

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

Roadmap

1. Background: Models of computation, Logical systems
2. Logic and Computation
3. Communication and Computation
4. Federated Learning (AI) and Computation

Roadmap

1. Background: Models of computation, Logical systems
2. Logic and Computation
3. Communication and Computation
4. Federated Learning (AI) and Computation

Models of computation

Expressiveness - Effective computability (mid 1930s)

- **(Turing)** Equivalence of Turing machines and λ-calculus
- **(Kleene)** Equivalence of Recursive functions and λ-calculus
- **(Curry)** Equivalence of Combinatory Logic and λ-calculus

λ-calculus - theory of functions

Syntax

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

Reduction rules

α-reduction:

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

β-reduction:

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

λ-calculus - theory of functions

Syntax

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

Reduction rules

α-reduction:

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

β-reduction:

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

Example

$(\lambda x.x)5 \to 5$ identity funtion

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

Logical systems

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

Roadmap

1. Background: Models of computation, Logical systems
2. Logic and Computation
3. Communication and Computation
4. Federated Learning (AI) and Computation

Curry - Howard correspondence

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

Intuitionistic logic vs Computation

$$\vdash A \quad \Leftrightarrow \vdash M : A$$

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 ✗

Proofs-as-programs for classical logic

CLASSICAL LOGIC	COMPUTATION
Axiomatic system	$\mathcal{C}$ calculus
Natural Deduction	$\lambda\mu$-calculus
Sequent system	$\overline{\lambda}\mu\widetilde{\mu}$-calculus

- Griffin, Felleisen, Filinsky 1990s (axiomatic)
  $\mathcal{C}$: *formulae-as-types notion of control,* `call/cc`
- Parigot 1992 (natural deduction)
  $\lambda\mu$: *algorithmic interpretation of classical logic*
- Curien, Herbelin 2000 (sequent)
  $\overline{\lambda}\mu\widetilde{\mu}$: *symmetric lambda calculus - duality of computation*

- Pierce's law is inhabited ✓

$$\lambda x.\mu\alpha. < x \mid (\lambda y.\mu\beta. < y \mid \alpha >) \bullet \alpha >: ((A \to B) \to A) \to A$$

Intuitionistic logics extended

PROP	proposition logic
PROP2	second-order proposition logic
PROP$\underline{\omega}$	weakly higher-order proposition logic
PROPω	higher-order proposition logic
PRED	predicate logic
PRED2	second-order predicate logic
PRED$\underline{\omega}$	weakly higher-order predicate logic
PREDω	higher-order predicate logic

Intuitionistic logics extended - Computation

PROP	proposition logic	$\lambda\rightarrow$
PROP2	second-order proposition logic	$\lambda 2$ ($\mathcal{F}$)
PROP$\underline{\omega}$	weakly higher-order proposition logic	$\lambda\underline{\omega}$
PROPω	higher-order proposition logic	$\lambda\omega$ ($\mathcal{F}\omega$)
PRED	predicate logic	λP
PRED2	second-order predicate logic	$\lambda P2$
PRED$\underline{\omega}$	weakly higher-order predicate logic	$\lambda P\underline{\omega}$
PREDω	higher-order predicate logic	$\lambda P\omega$ (CC)

$$
\begin{tikzcd}
 & \mathit{PROP}\omega/\lambda\omega \arrow[rr] & & \mathit{PRED}\omega/\lambda P\omega \\
\mathit{PROP}2/\lambda 2 \arrow[ur] \arrow[rr] & & \mathit{PRED}2/\lambda P2 \arrow[ur] & \\
 & \mathit{PROP}\underline{\omega}/\lambda\underline{\omega} \arrow[uu] \arrow[rr] & & \mathit{PRED}\underline{\omega}/\lambda P\underline{\omega} \arrow[uu] \\
\mathit{PROP}/\lambda\!\to \arrow[uu] \arrow[ur] \arrow[rr] & & \mathit{PRED}/\lambda P \arrow[uu] \arrow[ur] &
\end{tikzcd}
$$

(a) Logic and Lambda cube

$\vdash A$ (LOGIC) if and only if $\vdash M : A$ (λLOGIC)

Good properties of the cube

- Uniqueness of types ✓
- Confluence (Church-Rosser property) ✓
- Type preservation under reduction (Subject Reduction) ✓
- Termination (Strong Normalisation) ✓
- Expressivness

proofs-as-programs } → interactive proof assistant

COQ

$$
\begin{tikzcd}
 & PROP\omega/\lambda\omega \arrow[rr] & & PRED\omega/\lambda P\omega \\
PROP2/\lambda 2 \arrow[ur] \arrow[rr] & & PRED2/\lambda P2 \arrow[ur] & \\
 & PROP\underline{\omega}/\lambda\underline{\omega} \arrow[uu] \arrow[rr] & & PRED\underline{\omega}/\lambda P\underline{\omega} \arrow[uu] \\
PROP/\lambda\to \arrow[uu] \arrow[ur] \arrow[rr] & & PRED/\lambda P \arrow[uu] \arrow[ur] &
\end{tikzcd}
$$

(a) Logic and Lambda cube

① theorem proving - Fundamental thm. Algebra

② proof checking - 4 color thm.

③ verification - CompCert

$\vdash A$ (LOGIC) if and only if $\vdash M : A$ (λLOGIC)

Roadmap

1. Background: Models of computation, Logical systems
2. Logic and Computation
3. Communication and Computation
4. Federated Learning (AI) and Computation

Proofs-as-programs paradigm extended

COMPUTATION	COMMUNICATION
determinism	non-determinism
term	process
sequential composition	concurrency
computational behaviour	interactional behaviour
λ calculus	π calculus, CCS, CSP

formulae – as – types
proofs – as – terms
proofs – as – programs
proofs – as – processes

Good properties

- Types preservation (Subject Reduction) ✓
- Progress ✓
- Consequences
  - Safety = Preservation + Progress
  - Liveness
  - Deadlock freedom

Good properties

Typed programs cannot "go wrong"

- Types preservation (Subject Reduction) ✓
- Progress ✓
- Consequences
  - Safety = Preservation + Progress
  - Liveness
  - Deadlock freedom

Roadmap

1. Background: Models of computation, Logical systems
2. Logic and Computation
3. Communication and Computation
4. Federated Learning (AI) and Computation

Federated Learning + Formal Verification

- **Federated learning (FL)** a machine learning setting where clients keep training data decentralised and collaboratively train a model
- **Formal verification** a process of mathematically checking that the behaviour of a system satisfies a given property
  proofs-as-programs & proofs-as-processes

Sounds great ✓

but there is zero previous work to build upon ✗

Hence, we need to take small steps to find

a common language of the two working communities ??

Work in progress

- **Python Testbed for Federated Learning Algorithms** (PTB-FLA)
  - *under development at UNS*
- **Communicating Sequential Processes** calculus (CSP)
  - to model PTB-FLA
- **Process Analysis Toolkit model checker** (PAT)
  - to verify properties of the CSP models

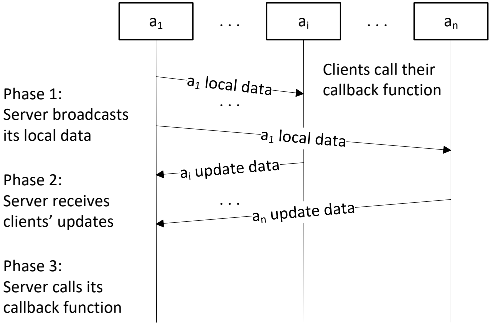
PTB-FLA

- Python Testbed for Federated Learning Algorithms (PTB-FLA)
- Developed with the primary intention to be used as **a framework for developing federated learning algorithms**
- It is a work in progress, and it supports both centralised and decentralised algorithms
  - the generic **centralised** one-shot FLA execution
  - the generic **decentralised** one-shot FLA execution

M. Popovic, M. Popovic, I. Kastelan, M. Djukic, S. G.: A simple python testbed for federated learning algorithms. In: ZINC 2023. pp. 148-153 (2023).

Centralised - star topology

- The algorithm goes in 3 phases where:
  - local data is the local machine learning model
  - private data is training data
- At this point our focus was on the communication pattern:
  - broadcasting
  - receiving from clients in any order!



Formal verification of FL protocols

Current approach:

- to use CSP (Communicating Sequential Processes) calculus to model (bottom up)
  - the **centralised** (star topology) FL protocol
  - the **decentralised** (clique topology) FL protocol
- to use PAT (Process Analysis Toolkit) model checker to prove
  - **deadlock freedom** and
  - **termination**

  of the two CSP models (top down)

Ongoing research:

- to automatise the translation of the Python code into the CSP model

Reference

I. Prokić, S. G., S. Kašterović, M. Popovic, M. Popovic, I. Kaštelan
Correct orchestration of Federated Learning generic algorithms: formalisation and verification in CSP
ECBS 2023 - Engineering of Computer-Based Systems
Lecture Notes in Computer Science 14390, pp 274–288 (2023)

On arXiv

Scan me!

www.project-tardis.eu

EU’s Horizont Europe

TaRDIS

Trustworthy and Resilient Decentralised Intelligence for Edge Systems

PROJECT-TARDIS.EU

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	???