The Curry-Howard correspondence journey

Silvia Ghilezan

University of Novi Sad
Mathematical Institute SASA, Serbia

Lecture 3

Proof Society International School
Birmingham, September 2024

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 3

Intersection types

Communication and Computation

Federated Learning (AI) and Computation

Roadmap

Intersection types

Communication and Computation

Federated Learning (AI) and Computation

In $\lambda \rightarrow$

$$\lambda x.xx : \textit{NO}$$

In $\lambda 2$

$$\lambda x : (\forall \alpha.\alpha).x(\sigma \rightarrow \tau)(x\,\sigma) : (\forall \alpha.\alpha) \rightarrow \tau$$

terms depending on terms	$\lambda \rightarrow$
terms depending on types	$\lambda 2$

Intersection types

Language

- Terms Λ

$$M ::= x \mid MM \mid \lambda x.M$$

- Types

$$\sigma ::= \alpha \mid \sigma \to \sigma \mid \sigma \cap \sigma$$

$\lambda\cap$ type system = $\lambda \to$ extended (à la Curry)

- Rules

$$\frac{\Gamma \vdash M : \sigma \cap \tau}{\Gamma \vdash M : \sigma}\,(elim\cap) \qquad \frac{\Gamma \vdash M : \sigma \cap \tau}{\Gamma \vdash M : \tau}\,(elim\cap) \qquad \frac{\Gamma, \vdash M : \sigma \quad \Gamma \vdash M : \tau}{\Gamma \vdash M : \sigma \cap \tau}\,(intr\cap)$$

$$\frac{\Gamma, \vdash M : \sigma \quad \sigma \leq \tau}{\Gamma \vdash M : \tau}\,(\leq)$$

Intersection types

Introduced in the late 1970s, to overcome the limitations of simple types

- Coppo, Dezani
- Pottinger
- Sallé

$$\dfrac{\dfrac{\dfrac{x : (\sigma \to \tau) \cap \sigma \vdash x : (\sigma \to \tau) \cap \sigma}{x : (\sigma \to \tau) \cap \sigma \vdash x : \sigma \to \tau}\,(elim\cap) \qquad \dfrac{x : (\sigma \to \tau) \cap \sigma \vdash x : (\sigma \to \tau) \cap \sigma}{x : (\sigma \to \tau) \cap \sigma \vdash x : \sigma}}{x : (\sigma \to \tau) \cap \sigma \vdash xx : \tau}\,(elim$$

$$\dfrac{x : (\sigma \to \tau) \cap \sigma \vdash xx : \tau}{\vdash \lambda x.xx : ((\sigma \to \tau) \cap \sigma) \to \tau}\,(intr \to)$$

Intersection types - SN

Complete characterization of strong normalization

Theorem

$$M \text{ is typable} \iff M \text{ is SN}$$

Proof.

Typability $\Longrightarrow$ SN

- reducibility method

SN $\Longrightarrow$ Typability

- typability of normal forms
- head subject expansion

Complete characterisation of different normalisation properties: solvable, weakly solvable terms and standardisation, among others

- 3 Type ?
  - Typability is undecidable
  - Type checking is undecidablte
  - Inhabitation is undecidable (1990s)

- Filter models
  built by means of intersection types, enable to prove **completeness** of type assignment

Theorem

$$\Gamma \vdash M : \sigma \iff \Gamma \models M : \sigma$$

Filter models became powerful tool for proving completeness of type assignment in different frameworks

Curry-Howard?

- Intersection types do not correspond to inuitionistic conjunction

$$\sigma \to \tau \to \sigma \cap \tau$$

intuitionistically provable

$$\vdash \lambda x.\lambda y. < x, y >: \sigma \to \tau \to \sigma \wedge \tau$$

not inhabited in $\lambda\cap$

$$\nvdash M : \sigma \to \tau \to \sigma \cap \tau$$

- Curry-Howard correspondence is broken

proofs do not correspond to terms

$$\frac{\Gamma \vdash M : \sigma \cap \tau}{\Gamma \vdash M : \tau}\ (elim\cap) \qquad \frac{\Gamma, \vdash M : \sigma \quad \Gamma \vdash M : \tau}{\Gamma \vdash M : \sigma \cap \tau}\ (intr\cap)$$

Logic?

There is no perfect world

Area of ongoing research

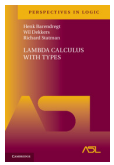
There have been several attempts

New ideas welcome

Several intersection typed calculi à la Church

F. Barbanera, M. Dezani-Ciancaglini, U. de'Liguoro, B. Venneri:
YACC: Yet Another Church Calculus.
Lecture Notes in Computer Science 14560: 17–35 (2024)

References



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

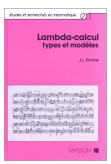
Lambda Calculus with Types.

Cambridge University Press, 2013.

H.P. Barendregt, M. Coppo, M. Dezani-Ciancaglini.

A filter lambda model and the completeness of type assignment.

Journal of Symbolic Logic 48(4):931–940, 1983



J.-L. Krivine.

Lambda-calcul, types et modèls

Masson 1990

Lambda calculus types and models, English translation

M. Dezani-Ciancaglini, S. Ghilezan, S. Likavec.

Behavioural inverse limit lambda-models.

Theoretical Computer Science 316(1): 49–74 (2004)

Roadmap

Intersection types

Communication and Computation

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

Process calculi, late 1970s

Formal language for describing patterns of interaction in concurrent systems

- Communicating Sequential Processes calculus (CSP), introduced by Tony Hoare
- Calculus of Communicating Systems (CCS), introduced by Robin Milner
- π-calculus, introduced by Robin Milner

Expressiveness

π - calculus

Processes

P, Q, M, N	$::=$	
	$x(y).P$	receive on x, bind it
	$\overline{x} < y > .P$	send y on channel, c
	$P \mid Q$	run P and Q in para
	$\nu x.P$	create new channel
	$!P$	replication
	$\mu x.P$	recursion
	if $M = N$ then P else Q	conditional
	0	null process

Reductions (communication)

$$\overline{x} < a > .P \mid x(y).Q \longrightarrow P \mid Q[y := a]$$

Types

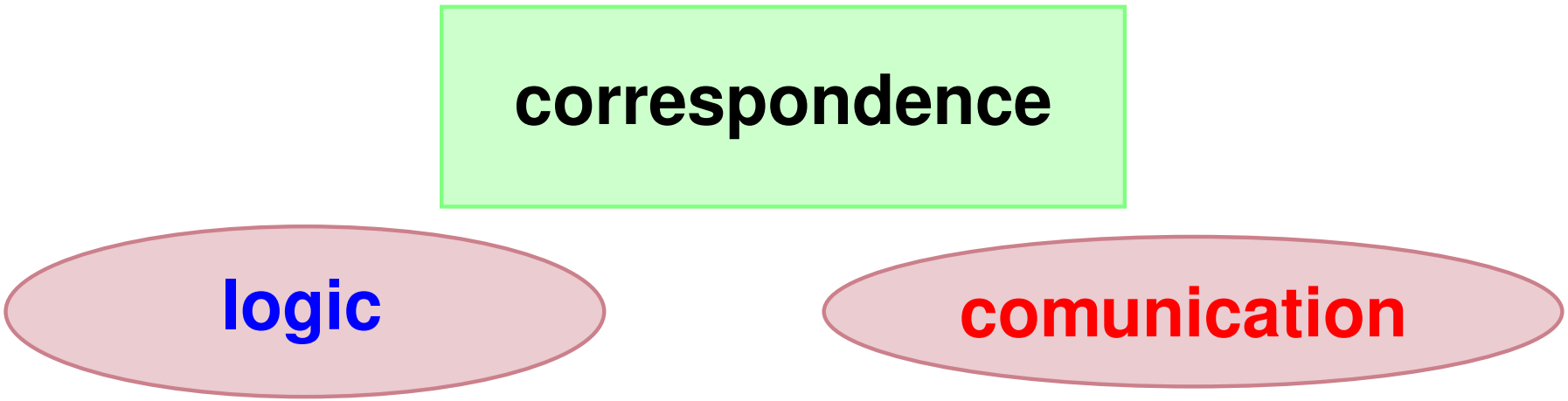
- Behavioural types for communication - centered systems ensure that the type system obeys the prescribed security policies (e.g., access control or secure information flow)
- Session types allow interactions to be structured into basic units, called sessions
- Multiparty session types
- The expressiveness of session types has enabled their application in diverse contexts, targeting
  - different programming models (functional and object-oriented programming)
  - operating system design
  - middleware communication protocols

Good properties

Typed programs cannot "go wrong"

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

Logic vs Communication



intuitionistic linear logic
sequent calculus

session types π-calculus

substructural logics in concurrency and programming

References

L. Caires, F. Pfenning, B. Toninho

Linear logic propositions as session types.

Mathematical Structures in Computer Science 26(3): 367-423 (2016)

C. A. R. Hoare

Communicating Sequential Processes

Prentice-Hall 1985 .

R. Milner

Communicating and Mobile Systems: The π - calculus

Cambridge University Press, 1999

D. Sangiorgi, D. Walker

The π - calculus

Cambridge University Press, 2003

S. Ghilezan, J. Pantovic, I. Prokic, A. Scalas, N. Yoshida.

Precise subtyping for asynchronous multiparty sessions

POPL 2021: 1-28 (2021)

Roadmap

Intersection types

Communication and Computation

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

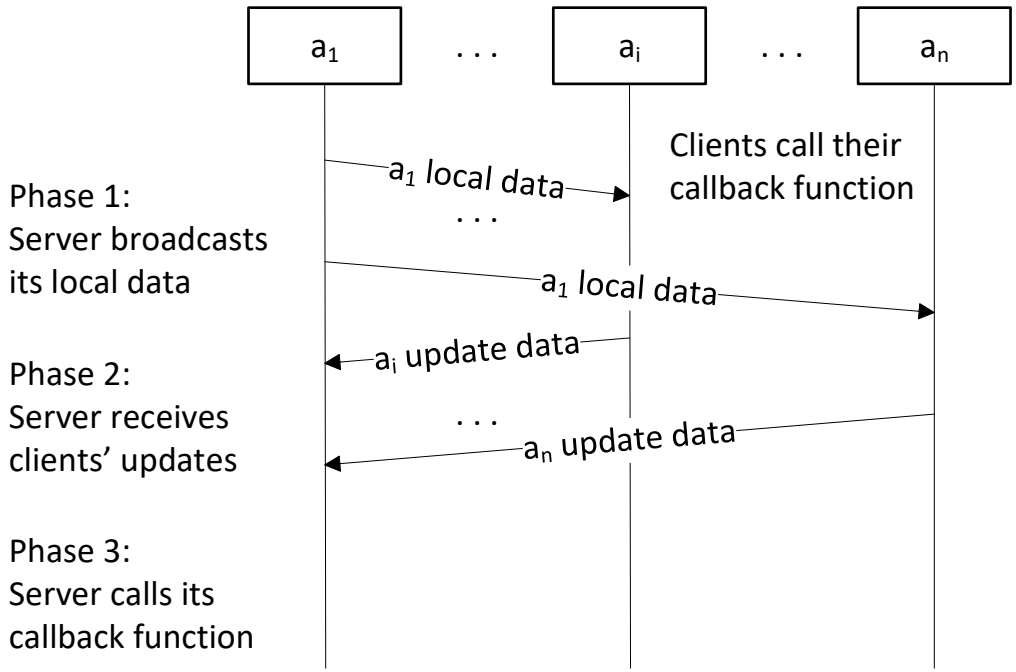
PTB-FLA

- Python Testbed for Federated Learning Algorithms (PTB-FLA)
- Developed with the primary intention to be used as **a framework for developing federated learning algorithms in small devices (swarm, smart grid)**
- 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).

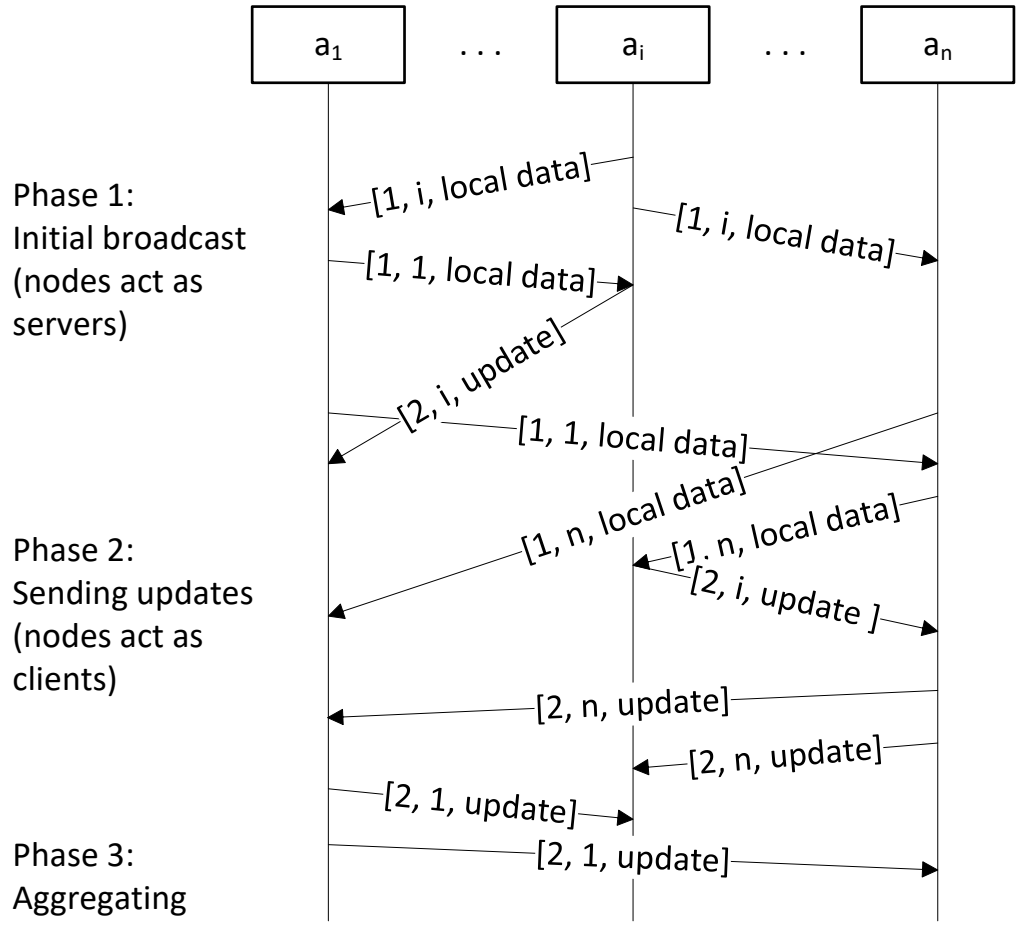
Centralised - star topology

- 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!



Decentralised - clique topology

- The algorithm goes in 3 phases where:
  - local data is 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!



Note: the term “update” in the messages above means “update data”.

Formal verification - proofs-as-programs

- Communicating Sequential Processes calculus (CSP) introduced by Tony Hoare
- Formal language for describing patterns of interaction in concurrent systems
- Process Analysis Toolkit (PAT) model checker
- Other process calculi: Calculus of Communicating Systems (CCS), π-calculus, introduced by Robin Milner

Formal verification of FL protocols

Current approach:

- **Formalisation**
  CSP (Communicating Sequential Processes, Hoare) models (bottom up approach)
  - the **centralised** (star topology) FL protocol
  - the **decentralised** (clique topology) FL protocol

- **Verification**
  PAT (Process Analysis Toolkit) model checker to prove
  - **deadlock freedom**
  - **termination**

  of the two CSP models (top down)

Ongoing research:

- to automatise the translation of the Python code into the CSP model
- multiparty session types - correct by construction

Model checking with PAT

Centralised

```
1 ####assert SysCentralised() deadlockfree;
2 ####define Terminated (terminated == True);
3 ####assert SysCentralised() reaches Terminated;
4 ####assert SysCentralised() |= []<> Terminated;
```

Decentralised

```
1 #assert SysDecentralised() deadlockfree;
2 #define Terminated (terminated == True);
3 #assert SysDecentralised() reaches Terminated;
4 #assert SysDecentralised() |= []<> Terminated;
```

I. Prokić, S. Ghilezan S., S. Prokić, M. Popovic, M. Popovic, I. Kastelan
Correct orchestration of Federated Learning generic algorithms: formalisation and verification in CSP
ECBS 2023 LNCS 14390: 274-288 (2023)

www.project-tardis.eu

EU's Horizont Europe

TaRDIS

Trustworthy and Resilient Decentralised Intelligence for Edge Systems

PROJECT-TARDIS.EU

Endless 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	???

Endless journey

Theory

- substructural logics
- modal logics
- probabilistic logics
- TBC

Application

- correctness by construction
- specification (software, hardware)
- verification (software, hardware)
- TBA - new technologies will name it

In lieu of conclusion

- reread
- revisit
- rethink
- rewrite