Session Type Systems Compared:

The Case of Deadlock Freedom

Ornela Dardha (University of Glasgow)
Jorge Pérez (University of Groningen)

TYPES 2022

Oilthigh Ghlaschu

π CoLab Concurrency Lab

About this work...

- Based on a recent paper published at J. Log. Algebraic Methods Program. (JLAMP) 2022.
- It investigates the deadlock freedom (DF) property in the context of message-passing concurrent processes.
- It provides a comparative study and expressiveness result of different type systems guaranteeing deadlock freedom.

The Curry-Howard Correspondence

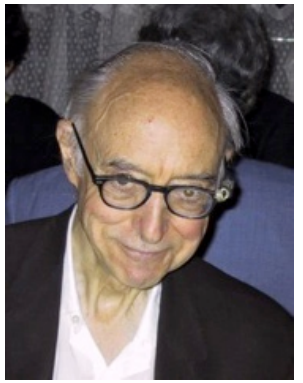
Deep correspondence between **types** and **logic**: *foundation of functional programming.*

Haskell Curry

types ≈ propositions

programs ≈ proofs

evaluation ≈ proof normalisation
(cut elimination)



William Howard

Example:
a function of type A ➔B *corresponds* to a proof of A **implies** B;
computationally, it constructs a proof of B (the result) from a proof of A (the parameter).

The Curry-Howard Correspondence

Intuitionistic Natural Deduction ≈ Simply Typed Lambda Calculus

Quantification over Propositions ≈ Polymorphism

Modal Logic ≈ Monads (state, exceptions)

??? ≈ Concurrent Computation

CH for concurrency?

- In 1987, Girard speculated that linear logic could form the basis of a Curry-Howard correspondence for concurrent computation.

"The new connectives of linear logic have obvious meanings in terms of parallel computation. […] Linear logic is the first attempt to solve the problem of parallelism at the logical level, i.e., by making the success of the communication process only dependent of the fact that the programs can be viewed as proofs of something, and are therefore sound."

-- Girard 1987

Jean-Yves Girard

CH for concurrency?

Connections between **linear logic** and the **pi-calculus** were developed [Abramsky 1990, 1994; Bellin & Scott 1994], but did not become *foundation of concurrent programming*.

Samson Abramsky

Gianluigi Bellin

Phil Scott

Session Types

- Session types were introduced by Honda et al. [1993, 1994, 1998] as type-theoretic specifications of communication protocols.

  ?A . B *receive* a message of type A, then *continue* protocol B.
  ! A . B *send* a message of type A, then *continue* protocol B.

- Duality: A and $A^{\perp}$ are complementary views of a protocol.

- During the subsequent years, session types have developed into a large and active research area.

Session Types and Linear Logic

Caires and Pfenning [2010] discovered a correspondence between **session types** for **pi-calculus** and dual intuitionistic **linear logic**.
Wadler [2012] correspondence with classical **linear logic**.

Proof normalisation (**cut elimination**) corresponds to **communication**.

Luís Caires

Frank Pfenning

Phil Wadler

Session Types and Linear Logic

session types ≈ propositions

pi-calculus processes ≈ proofs

communication ≈ proof normalisation
(cut elimination)

- ?A . B corresponds to $A \parr B$
- !A . B corresponds to $A \otimes B$
- Branch $\&\{l_i : A_i\}_{i \in I}$ and
- Select $\oplus\{l_i : A_i\}_{i \in I}$ are the same both in session types and linear logic

Pi calculus and session types

- The pi-calculus is a model of communication and concurrency, based on the notion of *processes,* which communicate over *channels*.
- Several type systems exist for the pi-calculus, most notably using
  - Linear channel types
  - Session types
  - Standard channel types
- Several type systems guarantee DF.
- The goal is to investigate and compare type systems for DF.
- We use (the CH correspondence with) session types as the ***yardstick*** for our comparison.

Pi-calculus by example: (no) deadlocks

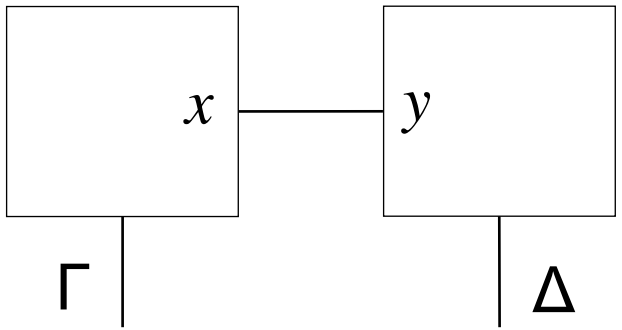
A deadlock arises from **cyclic dependency** between communication operations, when two processes share *at least two channels*.

$$(\nu x_1 y_1)(\nu x_2 y_2)\left[x_1(z).x_2(w).\mathbf{0} \mid y_1[42].y_2[\mathtt{true}].\mathbf{0}\right]$$ **OK**

$$(\nu x_1 y_1)(\nu x_2 y_2)\left[x_1(z).x_2(w).\mathbf{0} \mid y_2[\mathtt{true}].y_1[42].\mathbf{0}\right]$$ **STUCK**

DF: Linear Logic approach

- The linear logic type system guarantees deadlock freedom because *two processes can only be connected by a **single** channel*.
- The linear logic type system **rejects** both the processes in the previous slide because *they are connected by **two** channels*.



$$\frac{P \vdash \Gamma,\, x{:}A \qquad Q \vdash \Delta,\, y{:}A^{\perp}}{(\nu x^{A} y)(P \mid Q) \vdash \Gamma, \Delta}\ (\textsf{cut})$$

DF: Priority-based approach

- Kobayashi [1997-]; Padovani [2013, 2014] developed type systems for deadlock-free pi-calculus processes based on priorities.

- Priorities o, o'… are *natural numbers* used to annotate types.

- Priority-based type systems *type more processes* than linear logic, because they allow processes to share more than a single channel.

- Priorities must obey the following laws:

(i) an action (input/output) of priority o must be prefixed only by actions of priorities *strictly smaller* than o.

(ii) communication requires *equal* priorities of dual actions.

This work

We formally compare $\mathcal{L}$ and $\mathcal{K}$, two classes of DF (session) typed processes.

- $\mathcal{L}$ = session processes that are well typed under the Curry-Howard correspondence with Linear Logic.

- $\mathcal{K}$ = *standard* session processes that satisfy DF *and* use a priority-based type system à la Kobayashi.

This work

We develop two technical results.

- Separation: to separate $\mathcal{L}$ and $\mathcal{K}$, we define muK---a subclass of $\mathcal{K}$ whose definition internalizes the key aspects of CH correspondence with ST. Note: muK adopts a cut rule.

- Unification: to unify $\mathcal{L}$ and $\mathcal{K}$, we define two translations of processes in $\mathcal{K}$ into $\mathcal{L}$, by making the process in $\mathcal{K}$ 'more parallel'.

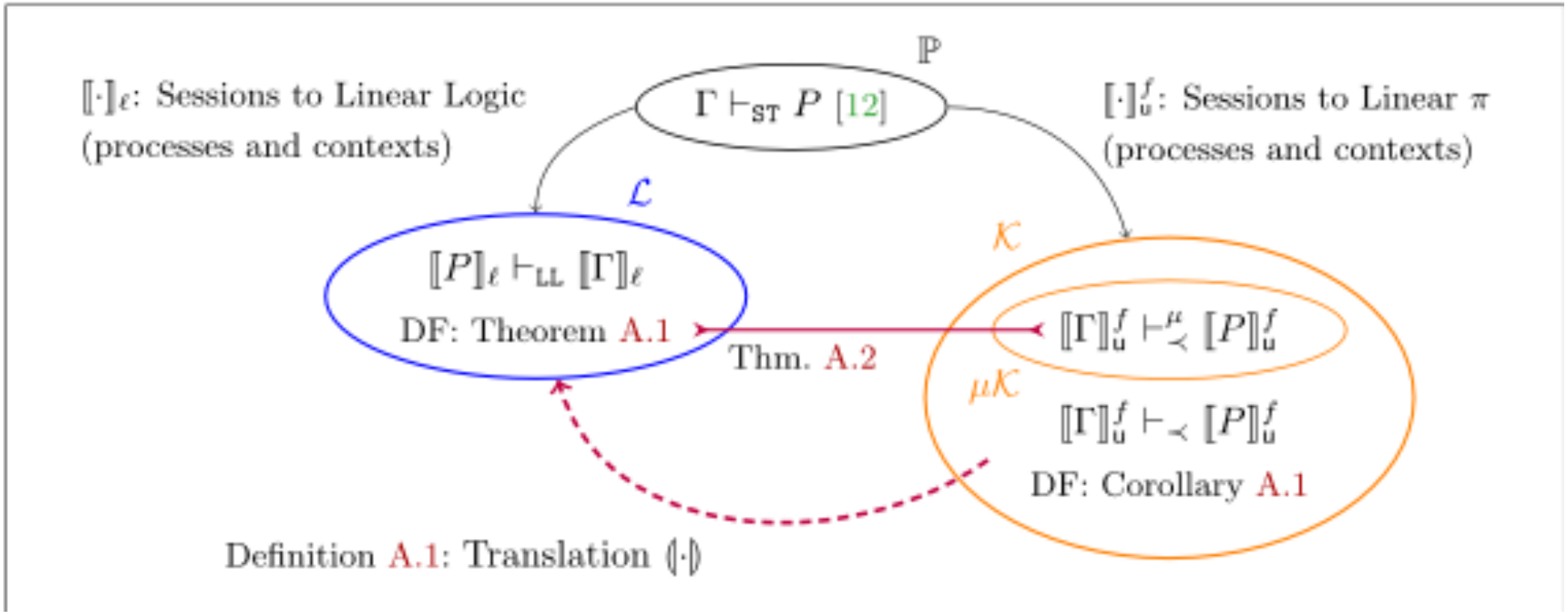
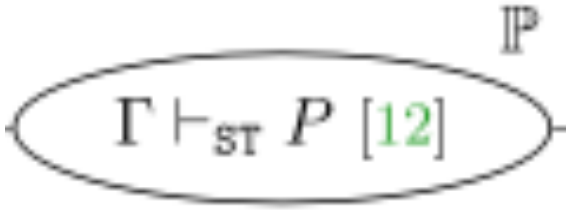
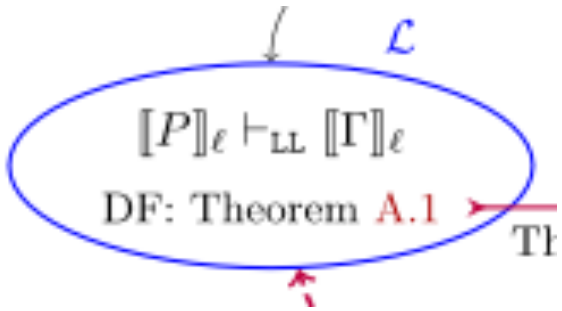

$\mathbb{P}$
$\Gamma \vdash_{\mathsf{ST}} P$ [12]
$[\![\cdot]\!]_\ell$: Sessions to Linear Logic
(processes and contexts)
$[\cdot]^f_u$: Sessions to Linear π
(processes and contexts)
$\mathcal{L}$
$\mathcal{K}$
$[\![P]\!]_\ell \vdash_{\mathsf{LL}} [\![\Gamma]\!]_\ell$
DF: Theorem A.1
Thm. A.2
$[\![\Gamma]\!]^f_u \vdash^{\mu}_{\prec} [P]^f_u$
$\mu\mathcal{K}$
$[\![\Gamma]\!]^f_u \vdash_{\prec} [P]^f_u$
DF: Corollary A.1
Definition A.1: Translation $(\!|\cdot|\!)$

Figure explained



- This typing judgment determines the class of processes typed by standard session types.
- [12] V. Vasconcelos. Fundamentals of session types. *Inf. Comput*. 217:52—70, 2012

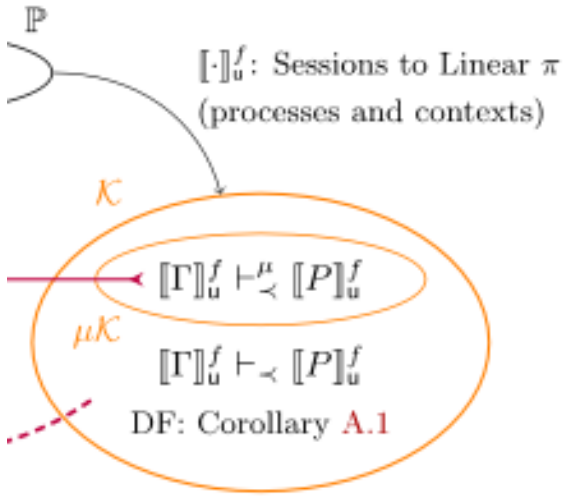
Figure explained



- We let live(P) be a predicate stating that process P has a pending input/output action to take.

Theorem A.1 (Deadlock Freedom). *If* $P \vdash_{\mathrm{LL}} \cdot$ *and* $live(P)$ *then* $P \longrightarrow Q$, *for some* Q.

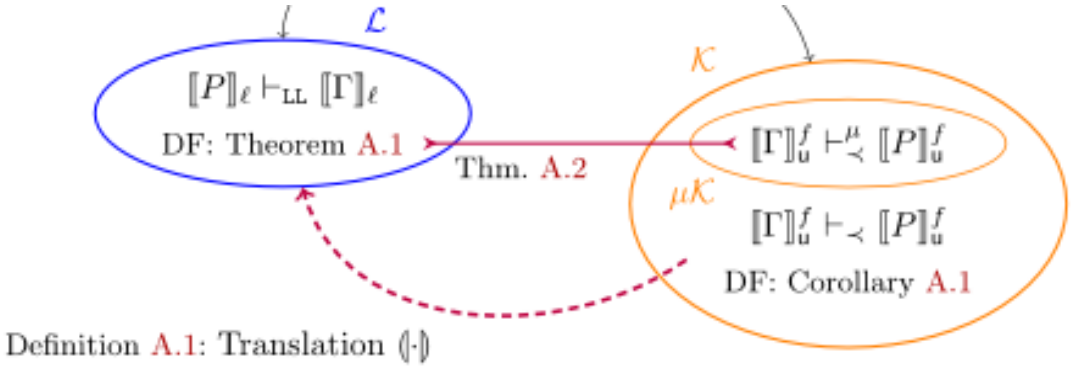
Figure explained



- Encoding session types to linear types: OD. E.Giachino, D.Sangiorgi. Session Types Revisited. *PPDP*, 2012

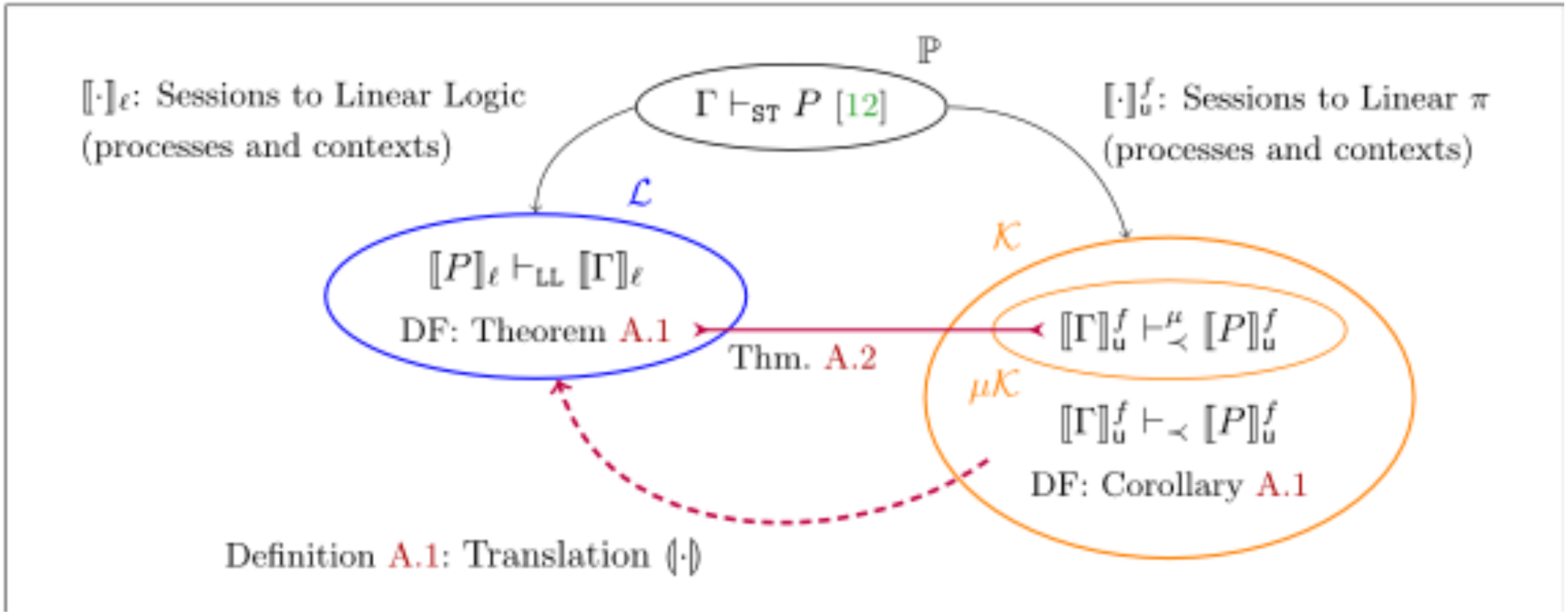
Corollary A.1. *Let* $\vdash_{\mathsf{ST}} P$ *be a session process. If* $\vdash_{\prec} [\![P]\!]^f_{\mathfrak{u}}$ *is deadlock-free then* P *is deadlock-free.*

Figure explained



- Separation: **Theorem A.2.** $\mathcal{L} = \mu\mathcal{K}$.
- Unification: **Definition A.1** (Translation into $\mathcal{L}$).

- The translation is *type preserving* and *operationally correspondent*.


$\mathbb{P}$
$\Gamma \vdash_{\mathsf{ST}} P$ [12]
$[\![\cdot]\!]_\ell$: Sessions to Linear Logic
(processes and contexts)
$[\cdot]^f_u$: Sessions to Linear π
(processes and contexts)
$\mathcal{L}$
$\mathcal{K}$
$[\![P]\!]_\ell \vdash_{\mathsf{LL}} [\![\Gamma]\!]_\ell$
DF: Theorem A.1
Thm. A.2
$[\![\Gamma]\!]^f_u \vdash^{\mu}_{\prec} [P]^f_u$
$\mu\mathcal{K}$
$[\![\Gamma]\!]^f_u \vdash_{\prec} [P]^f_u$
DF: Corollary A.1
Definition A.1: Translation $(\!|\cdot|\!)$

Thank you!