University of Glasgow | School of Computing Science

EPSRC
Engineering and Physical Sciences Research Council

A New Linear Logic for Deadlock-Free Session-Typed Processes

Ornela Dardha
(joint work with Simon Gay)

University of Leicester
23 November 2018

The Curry-Howard Correspondence

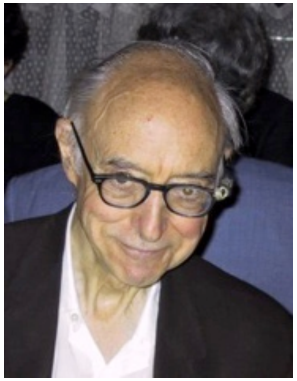
The deep correspondence between **types** and **logic**:
foundation of functional programming.

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).

Curry-Howard for Concurrency?

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

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

Jean-Yves Girard

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 <u>receive</u> a message of type <u>A</u>, then <u>continue</u> protocol <u>B</u>.
! A . B <u>send</u> a message of type <u>A</u>, then <u>continue</u> protocol <u>B</u>.

<u>Duality</u>: **A** and $\mathbf{A}^{\perp}$ are complementary views of a protocol.

During the subsequent 20+ years, session types developed into a large and active research area.

Kohei Honda

Curry-Howard for Session Types

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

The logical approach to session types has been extended: dependent types, failures, sharing and races,…

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

Luís Caires

Frank Pfenning

Phil Wadler

Further work by Caires *et al*. [2010 -]; Wadler [2012 -] …

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...

Input corresponds to **par**

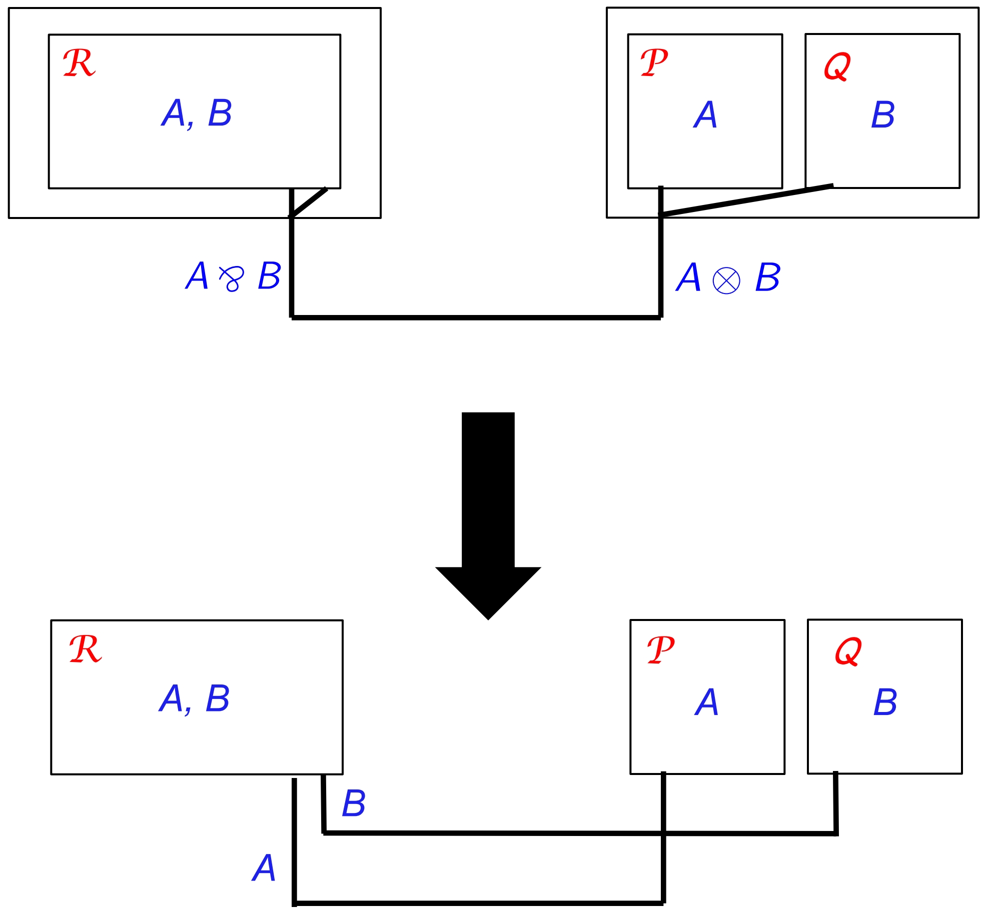
$$\frac{R \vdash \Theta, y : A, x : B}{x(y).R \vdash \Theta, x : A \parr B} \parr$$

Output corresponds to **tensor**

$$\frac{P \vdash \Gamma, y : A \quad Q \vdash \Delta, x : B}{x[y].(P \mid Q) \vdash \Gamma, \Delta, x : A \otimes B} \otimes$$

Notice the threading of the continuation channel through the rules.

Cut Elimination as Communication



Pi-Calculus: Syntax of Processes

$$
\begin{array}{rlr}
P, Q ::= & x[y].P & \text{(output)} \\
& x(y).P & \text{(input)} \\
& x \triangleleft l_j.P & \text{(selection)} \\
& x \triangleright \{l_i : P_i\}_{i \in I} & \text{(branching)} \\
& \mathbf{0} & \text{(inaction)} \\
& P \mid Q & \text{(composition)} \\
& (\boldsymbol{\nu} x^A y)P & \text{(session restriction)}
\end{array}
$$

Deadlock in Communicating Systems

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

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

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

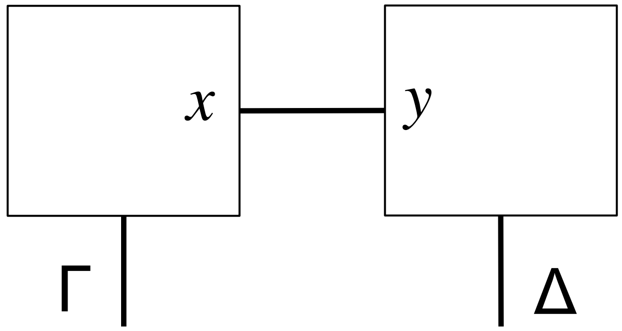
Deadlock-Freedom and Linear Logic

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

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

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

The linear logic type system guarantees deadlock-freedom because *two processes can only be connected by a **single** channel*.



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

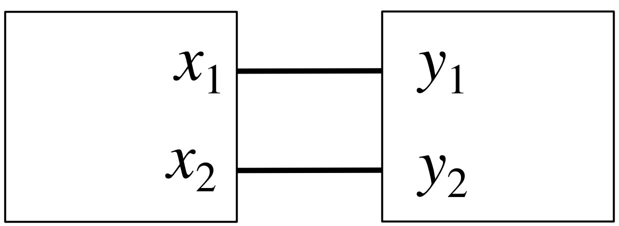
Deadlock-Freedom and Linear Logic

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

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

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

The linear logic type system **rejects** both the processes above because *they are connected by* ***two*** *channels*.



?

Priority-based Typing for Deadlock-Freedom

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

Priorities **o**, **o'**… are natural numbers and annotate **types**.

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.

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

Exercise: are the following processes typabable?

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

OK
typable

(i) $\mathrm{pr}(x_1) < \mathrm{pr}(x_2)$ $\mathrm{pr}(y_1) < \mathrm{pr}(y_2)$
(ii) $\mathrm{pr}(x_1) = \mathrm{pr}(y_1)$ $\mathrm{pr}(x_2) = \mathrm{pr}(y_2)$

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

STUCK
untypable

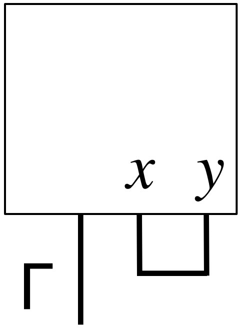
(i) $\mathrm{pr}(x_1) < \mathrm{pr}(x_2)$ $\mathrm{pr}(y_2) < \mathrm{pr}(y_1)$
(ii) $\mathrm{pr}(x_1) = \mathrm{pr}(y_1)$ $\mathrm{pr}(x_2) = \mathrm{pr}(y_2)$

Our Goal: Priority-based Linear Logic Typing

We combine classical **linear logic** with **priorities** for a more expressive session type system for deadlock-free processes.

Replace the cut rule with mix and cycle.

$$\text{(mix)}\ \frac{P \vdash \Gamma \qquad Q \vdash \Delta}{P \mid Q \vdash \Gamma, \Delta} \qquad\qquad \text{(cycle)}\ \frac{P \vdash \Gamma, x : A, y : A^{\perp}}{(\boldsymbol{\nu} x^{A} y) P \vdash \Gamma}$$



With these rules, multicut is derivable.

$$\text{(multicut)}\ \frac{P \vdash \Gamma, x_1 : A_1, \ldots, x_n : A_n \qquad Q \vdash \Delta, y_1 : A_1^{\perp}, \ldots, y_n : A_n^{\perp}}{(\boldsymbol{\nu} x_1^{A_1} y_1 \ldots x_n^{A_n} y_n)(P \mid Q) \vdash \Gamma, \Delta}$$

Our Goal: Priority-based Linear Logic Typing

Propositions are annotated with priorities.

$$A, B ::= \bot^{o} \mid \mathbf{1}^{o} \mid A \otimes^{o} B \mid A \parr^{o} B \mid \oplus^{o} \{l_i : A_i\}_{i \in I} \mid \&^{o}\{l_i : A_i\}_{i \in I} \mid ?^{o} A \mid !^{o} A$$

Proofs must obey laws (i) and (ii) on priorities.
$\mathsf{o} < \mathsf{pr}(\Gamma)$ satisfies law (i) on *strictly smaller*.

$$\frac{P \vdash \Gamma, y : A, x : B \quad \mathsf{o} < \mathsf{pr}(\Gamma)}{x(y).P \vdash \Gamma,\ x : A \parr^{o} B}\ (\parr) \qquad \frac{P \vdash \Gamma,\ y : A,\ x : B \quad \mathsf{o} < \mathsf{pr}(\Gamma)}{x[y].P \vdash \Gamma,\ x : A \otimes^{o} B}\ (\otimes)$$

$$\frac{P \vdash ?\Gamma,\ y : A \quad \mathsf{o} < \mathsf{pr}(\Gamma)}{!x(y).P \vdash ?\Gamma,\ x : !^{o} A}\ (!) \qquad \frac{P \vdash \Gamma,\ y : A \quad \mathsf{o} < \mathsf{pr}(\Gamma)}{?x[y].P \vdash \Gamma,\ x : ?^{o} A}\ (?)$$

Our Goal: Priority-based Linear Logic Typing

When forming a cycle, the priorities of the connected types must be *equal*, to satisfy (ii):

$$\frac{P \vdash \Gamma,\ x : A,\ y : A^{\perp}}{(\boldsymbol{\nu} x^{A} y) P \vdash \Gamma}\ (\text{cycle})$$

Meaning: eventually x and y will be ready to communicate at the same time/step, allowing a reduction step and a proof rewrite.

Equality of priorities is captured by the **duality** definition:

$$\begin{aligned}
(A \mathbin{⅋}^{\mathsf{o}} B)^{\perp} &= A^{\perp} \otimes^{\mathsf{o}} B^{\perp} & (\perp^{\mathsf{o}})^{\perp} &= \mathbf{1}^{\mathsf{o}} \\
(A \otimes^{\mathsf{o}} B)^{\perp} &= A^{\perp} \mathbin{⅋}^{\mathsf{o}} B^{\perp} & (\mathbf{1}^{\mathsf{o}})^{\perp} &= \perp^{\mathsf{o}} \\
(\&^{\mathsf{o}}\{l_i : A_i\}_{i \in I})^{\perp} &= \oplus^{\mathsf{o}}\{l_i : A_i{}^{\perp}\}_{i \in I} & ?^{\mathsf{o}} A^{\perp} &= !^{\mathsf{o}} A^{\perp} \\
(\oplus^{\mathsf{o}}\{l_i : A_i\}_{i \in I})^{\perp} &= \&^{\mathsf{o}}\{l_i : A_i{}^{\perp}\}_{i \in I} & !^{\mathsf{o}} A^{\perp} &= ?^{\mathsf{o}} A^{\perp}
\end{aligned}$$

Beta-Reduction using Mix and Cycle

Beta-reduction for tensor/output and par/input: derivation

$$\dfrac{\dfrac{\dfrac{\mathsf{o} < \mathsf{pr}(\Gamma) \quad P \vdash \Gamma, v:A, x:B}{x[v].P \vdash \Gamma, x:A \otimes^{\mathsf{o}} B}(\otimes) \quad \dfrac{\mathsf{o} < \mathsf{pr}(\Delta) \quad Q \vdash \Delta, w:A^{\perp}, y:B^{\perp}}{y(w).Q \vdash \Delta, y:A^{\perp} \invamp^{\mathsf{o}} B^{\perp}}(\invamp)}{x[v].P \mid y(w).Q \vdash \Gamma, \Delta, x:A \otimes^{\mathsf{o}} B, y:A^{\perp} \invamp^{\mathsf{o}} B^{\perp}}(\mathsf{mix})}{(\boldsymbol{\nu} x^{A \otimes^{\mathsf{o}} B} y)(x[v].P \mid y(w).Q) \vdash \Gamma, \Delta}(\mathsf{cycle}) \quad \Longrightarrow \quad \dfrac{\dfrac{\dfrac{P \vdash \Gamma, v:A, x:B \quad Q \vdash \Delta, w:A^{\perp}, y:B^{\perp}}{P \mid Q \vdash \Gamma, \Delta, v:A, x:B, w:A^{\perp}, y:B^{\perp}}(\mathsf{mix})}{(\boldsymbol{\nu} x^{B} y)(P \mid Q) \vdash \Gamma, \Delta, v:A, w:A^{\perp}}(\mathsf{cycle})}{(\boldsymbol{\nu} v^{A} w)(\boldsymbol{\nu} x^{B} y)(P \mid Q) \vdash \Gamma, \Delta}(\mathsf{cycle})$$

Beta-reduction for other connectives: summary

$$(\boldsymbol{\nu} y^{A} z)(x \to y^{A} \mid P) \vdash \Gamma, x:A^{\perp} \implies P[x/z] \vdash \Gamma, x:A^{\perp}$$

$$(\boldsymbol{\nu} x^{A} y)(x[\,].\mathbf{0} \mid y().P) \vdash \Gamma \implies P \vdash \Gamma$$

$$(\boldsymbol{\nu} x^{\oplus^{\mathsf{o}}\{l_i:B_i\}_{i\in I}} y)(x \triangleleft l_j.P \mid y \triangleright \{l_i : Q_i\}_{i\in I}) \vdash \Gamma, \Delta \implies (\boldsymbol{\nu} x^{B_j} y)(P \mid Q_j) \vdash \Gamma, \Delta$$

$$(\boldsymbol{\nu} x^{!^{\mathsf{o}} A} y)(!x(v).P \mid ?y[w].Q) \vdash ?\Gamma, \Delta \implies (\boldsymbol{\nu} v^{A} w)(P \mid Q) \vdash ?\Gamma, \Delta$$

Priority-based Classical Linear Logic: Results

Theorem (cycle elimination). Given a proof of a sequent, we can construct a *cycle-free* proof for it.

Proof: (following cut elimination proof) a cycle is eliminated by either:

i) replacing it with another cycle on smaller propositions;

ii) pushing it further up the proof tree.

Since we are allowing cyclic structures,

how do we make sure we capture only the good ones?

Kobayashi's Reasoning of Deadlock-Freedom

A concurrent system is a collection of parallel processes, each with a top-level input or output action (prefix).

Pick a top-level prefix with *smallest* priority **o**: $x(z)$, say.

Somewhere there is the co-action $y[42]$ with *equal* priority **o**.

$y[42]$ must be in a **different parallel component**, otherwise it would be guarded by $x(z)$, requiring **o** < **o**.

$y[42]$ must be a **top-level prefix**, otherwise it is guarded by a prefix with priority **o'** < **o**, contradicting the dominance of **o**.

Communication on endpoints x and y is possible immediately.

Priority-based CP: Results

Cycle elimination corresponds to **communication**.

Theorem (subject reduction). Well-typed processes reduce to well-typed processes.

Proof: beta-reductions and commuting conversions.

Theorem (top-level deadlock freedom). If process P is well typed and it is a cycle, then there is some Q, such that P reduces to Q and Q is not a cycle.

Proof: follows from cycle elimination.

Conclusion and Future Work

- Presented a new priority-based linear logic combining mix and cycle rules with Kobayashi's priorities.
- Used it as a basis for a Curry-Howard isomorphism with session typed pi-calculus, allowing "good" cyclic processes.
- We prove the cycle elimination theorem, obtaining as a result deadlock freedom for session typed processes.

- Future work:

i) develop a type system for a functional language, GV with cycle and translate it to our system.

ii) extend our priority-based logic to allow recursion and sharing.

Thank you!

Questions?