Wen Kokke

FORWARDERS SHOULD BE LAZY

A TALK ABOUT A TINY DETAIL OF CLASSICAL PROCESSES

INTRODUCING CLASSICAL PROCESSES

What if Classical Linear Logic
was the type system for a process calculus?

$$
\begin{array}{rcllll}
A, B, C, D & := & A \otimes B & \mid & A \parr B & \text{delegation} \\
& \mid & A \oplus B & \mid & A \mathbin{\&} B & \text{choice} \\
& \mid & \forall X.\, A & \mid & \exists X.\, A & \text{polymorphism} \\
& \mid & \ldots & & & \text{etcetera}
\end{array}
$$

INTRODUCING CLASSICAL PROCESSES

What if Classical Linear Logic
was the type system for a process calculus?

stuff that receives

$$
\begin{array}{llcll}
A, B, C, D & := & A \otimes B & \mid \; A \parr B & \text{delegation} \\
 & \mid & A \oplus B & \mid \; A \mathbin{\&} B & \text{choice} \\
 & \mid & \forall X.\, A & \mid \; \exists X.\, A & \text{polymorphism} \\
 & \mid & \ldots & & \text{etcetera}
\end{array}
$$

stuff that sends

INTRODUCING CLASSICAL PROCESSES

What if Classical Linear Logic
was the type system for a process calculus?

$$
\begin{array}{lcll}
\overline{A \otimes B} & = & \overline{A} \parr \overline{B} & \text{delegation} \\
\overline{A \oplus B} & = & \overline{A} \mathbin{\&} \overline{B} & \text{choice} \\
\overline{\forall X.\, A} & = & \exists X.\, \overline{A} & \text{polymorphism} \\
\ldots & & & \text{etcetera}
\end{array}
$$

INTRODUCING CLASSICAL PROCESSES

What if Classical Linear Logic
was the type system for a process calculus?

$$
\begin{array}{lcll}
\overline{A} \otimes \overline{B} & = & \overline{A \parr B} & \text{delegation} \\
\overline{A} \oplus \overline{B} & = & \overline{A \mathbin{\&} B} & \text{choice} \\
\forall X.\, \overline{A} & = & \overline{\exists X.\, A} & \text{polymorphism} \\
\ldots & & & \text{etcetera}
\end{array}
$$

INTRODUCING CLASSICAL PROCESSES

$$
\begin{array}{rclcll}
P, Q, R & := & x{\leftrightarrow}y & & & \text{forwarder} \\
& \mid & (\nu x\bar{x})P & & & \text{new channel creation} \\
& \mid & P \parallel Q & & & \text{parallel composition} \\
& \mid & x[y].\,P & \mid & x(y).\,P & \text{send/receive delegation} \\
& \mid & x \triangleleft \ell.\,P & \mid & x \triangleright \{\ell : P_\ell\}_{\ell \in L} & \text{send/receive choice} \\
& \mid & x[A].\,P & \mid & x(A).\,P & \text{send/receive type} \\
& \mid & \ldots & & & \text{etcetera}
\end{array}
$$

(Disclaimer: This is technically Hypersequent Classical Processes. Potato, Tomato.)

INTRODUCING CLASSICAL PROCESSES

$$
\begin{array}{llll}
P, Q, R & := & x \leftrightarrow y & & \text{forwarder} \\
 & \mid & (\nu x \bar{x}) P & & \text{new channel creation} \\
 & \mid & P \parallel Q & & \text{parallel composition} \\
 & \mid & x[y].\, P & \mid \quad x(y).\, P & \text{send/receive delegation} \\
 & \mid & x \triangleleft \ell.\, P & \mid \quad x \triangleright \{\ell : P_\ell\}_{\ell \in L} & \text{send/receive choice} \\
 & \mid & x[A].\, P & \mid \quad x(A).\, P & \text{send/receive type} \\
 & \mid & \ldots & & \text{etcetera}
\end{array}
$$

(Disclaimer: This is technically Hypersequent Classical Processes. Potato, Tomato.)

INTRODUCING CLASSICAL PROCESSES

$$\frac{}{x{\leftrightarrow}y \vdash x : A, y : \overline{A}}\ (\text{Axiom})$$

$$\frac{P \vdash \Gamma \qquad Q \vdash \Delta}{P \parallel Q \vdash \Gamma \parallel \Delta}\ (\text{Branch})$$

$$\frac{P \vdash \Gamma, x : A \parallel \Delta, \bar{x} : \overline{A}}{(\nu x\bar{x})P \vdash \Gamma, \Delta}\ (\text{Cut})$$

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

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

(Disclaimer: This is technically Hypersequent Classical Processes. Potato, Tomato.)

INTRODUCING CLASSICAL PROCESSES

$$\frac{}{x{\leftrightarrow}y \vdash x : A, y : \overline{A}}\ (\text{Axiom})$$

$$\frac{P \vdash \Gamma \qquad Q \vdash \Delta}{P \parallel Q \vdash \Gamma \parallel \Delta}\ (\text{Branch})$$

$$\frac{P \vdash \Gamma, x : A \parallel \Delta, \bar{x} : \overline{A}}{(\nu x \bar{x})P \vdash \Gamma, \Delta}\ (\text{Cut})$$

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

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

(Disclaimer: This is technically Hypersequent Classical Processes. Potato, Tomato.)

INTRODUCING CLASSICAL PROCESSES

(Forward)

$$(\nu x\bar{x})(x{\leftrightarrow}w \parallel P)$$

does not need to be ready on $\bar{x}$!

$$\downarrow$$

$$P\{w/\bar{x}\}$$

(Delegate)

$$(\nu x\bar{x})(x[y].\,P \parallel \bar{x}(\bar{y}).\,Q)$$

$$\downarrow$$

$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel Q)$$

(Choose)

$$(\nu x\bar{x})(x \triangleleft \mathrm{inl}.\,P \parallel \bar{x} \triangleright \{\mathrm{inl} : Q; \mathrm{inr} : R\})$$

$$\downarrow$$

$$(\nu x\bar{x})(P \parallel Q)$$

(Instantiate)

$$(\nu x\bar{x})(x[A].\,P \parallel \bar{x}(X).\,Q)$$

$$\downarrow$$

$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel Q)$$

(Disclaimer: This is technically Hypersequent Classical Processes. However, the reduction rules are the same, so you cannot really tell.)

INTRODUCING CLASSICAL PROCESSES

(Forward)

$$(\nu x\bar{x})(x{\leftrightarrow}w \parallel P)$$
$$\downarrow$$
$$P\{w/\bar{x}\}$$

(Delegate)

$$(\nu x\bar{x})(x[y].\,P \parallel \bar{x}(\bar{y}).\,Q)$$
$$\downarrow$$
$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel Q)$$

(Choose)

$$(\nu x\bar{x})(x \triangleleft \mathrm{inl}.\,P \parallel \bar{x} \triangleright \{\mathrm{inl} : Q; \mathrm{inr} : R\})$$
$$\downarrow$$
$$(\nu x\bar{x})(P \parallel Q)$$

(Instantiate)

$$(\nu x\bar{x})(x[A].\,P \parallel \bar{x}(X).\,Q)$$
$$\downarrow$$
$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel Q)$$

(Disclaimer: This is technically Hypersequent Classical Processes. However, the reduction rules are the same, so you cannot really tell.)

INTRODUCING CLASSICAL PROCESSES

(Forward)

$$(\nu x\bar{x})(x \leftrightarrow w \parallel P)$$

$$\downarrow$$

$$P\{w/\bar{x}\}$$

(Delegate)

$$(\nu x\bar{x})(x[y].\, P \parallel \bar{x}(\bar{y}).\, Q)$$

$$\downarrow$$

$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel Q)$$

This is **asynchronous**.

This is **synchronous**.

Everything else is.

(Disclaimer: This is technically Hypersequent Classical Processes. However, the reduction rules are the same, so you cannot really tell.)

OH NO, IS THAT BAD?

Not really, but…

It complicates the metatheory a bunch.

It invalidates the simplest process interpretation.

It does a third thing so this list has three items?

(Disclaimer: I have not yet determined the third thing.)

IT COMPLICATES THE METATHEORY A BUNCH

It leads to a lot of special cases for forwarders...

A process is in canonical form when it does not contain (1) dual ready actions on the same channel or (2) any ready forwarder.

A process is in canonical form when all ready actions are blocked on external channels in the absence of ready forwarders.

IT INVALIDATES THE SIMPLEST PROCESS INTERPRETATION

What does this reduction rule require of an implementation?

$$\begin{array}{c}(\text{Forward})\\ (\nu x\bar{x})(x{\leftrightarrow}w \parallel P)\\ \downarrow\\ P\{w/\bar{x}\}\end{array}$$

The process P isn't required to be listening on $\bar{x}$.
This cannot be implemented as message-passing

(Disclaimer: There really wasn't a third thing. I am sorry for deceiving you.)

WHAT CAN WE DO?

WHAT DO? ① MAKE IT SYNCHRONOUS

(Forward)

$$(\nu x\bar{x})(x{\leftrightarrow}w \parallel P)$$

$$\downarrow$$

$$P\{w/\bar{x}\}$$

but...

only if $\mathbf{ready}(P, \bar{x})$

Simplifies the metatheory!

Simplifies the implementation...

A little bit...

WHAT DO? ② IDENTITY EXPANSION

Let's use Identity Expansion!

Identity Expansion is the dual of Cut Elimination.

It rewrites uses of the axiom to uses of the axiom with smaller formulas.

$$\frac{}{\vdash A \otimes B, \overline{A} \parr \overline{B}}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\dfrac{}{\vdash A, \overline{A}} \qquad \dfrac{}{\vdash B, \overline{B}}}{\vdash A \otimes B, \overline{A}, \overline{B}}}{\vdash A \otimes B, \overline{A} \parr \overline{B}}$$

WHAT DO? ② IDENTITY EXPANSION

On process, it rewrites forwarders to processes that explicitly do the forwarding.

But...

It is defined by recursion on the types of the endpoints—written over the arrow.

(why is that bad, Wen?

$$y \overset{A \otimes B}{\leftrightarrow} x,\ x \overset{A ⅋ B}{\leftrightarrow} y$$

$$\triangleq$$

$$x(z).\ y[w].\ (z \overset{A}{\leftrightarrow} w \parallel x \overset{B}{\leftrightarrow} y)$$

WHAT DO? ② IDENTITY EXPANSION

On process, it rewrites forwarders to processes that explicitly do the forwarding.

But...

It is defined by recursion on the types of the endpoints—written over the arrow.

$$y \overset{A\otimes B}{\leftrightarrow} x,\ x \overset{A⅋B}{\leftrightarrow} y$$

$$\triangleq$$

$$x(z).\ y[w].\ (z \overset{A}{\leftrightarrow} w \parallel x \overset{B}{\leftrightarrow} y)$$

WHAT DO? ③ MAKE IT LAZY

Expand the forwarder

lazily

in response to the kind of message received.

$$(\nu x\bar{x})(x[y].\,P \parallel \bar{x}{\leftrightarrow}w)$$

$$\triangleq$$

$$(\nu x\bar{x})(x[y].\,P \parallel \bar{x}(\bar{y}).\,w[z].\,(\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

$$\downarrow$$

$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel w[z].\,(\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

WHAT DO? ③ MAKE IT LAZY

$$(\nu x\bar{x})(x[y].\, P \parallel \bar{x}{\leftrightarrow}w)$$

Expand the forwarder

lazily

in response to the kind of message received.

$$\downarrow$$

$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel w[z].\, (\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

WHAT DO? ③ MAKE IT LAZY

Expand the forwarder

lazily

in response to the kind of message received.

$$(\nu x \bar{x})(x[y].\, P \parallel \bar{x} {\leftrightarrow} w)$$

$$\downarrow$$

$$(\nu x \bar{x})(\nu y \bar{y})(P \parallel w[z].\, (\bar{y} {\leftrightarrow} z \parallel \bar{x} {\leftrightarrow} w))$$

WHAT DO? ③ MAKE IT LAZY

Expand the forwarder

lazily

in response to the kind of message received.

(Forward-Delegate)

$$(\nu x\bar{x})(x[y].\, P \parallel \bar{x}{\leftrightarrow}w)$$

$$\downarrow$$

$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel w[z].\, (\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

WHAT DO? ③ MAKE IT LAZY

Expand the forwarder

lazily

in response to the kind of message received.

But...
Does this work?

(Forward-Delegate)

$$(\nu x\bar{x})(x[y].\,P \parallel \bar{x}{\leftrightarrow}w)$$

$$\downarrow$$

$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel w[z].\,(\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

WHAT DO? ③ MAKE IT LAZY

(Forward-Delegate)

$$(\nu x\bar{x})(x[y].\,P \parallel \bar{x}{\leftrightarrow}w)$$

$$\triangleq$$

$$(\nu x\bar{x})(x[y].\,P \parallel \bar{x}(\bar{y}).\,w[z].\,(\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

$$\downarrow$$

$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel w[z].\,(\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

(Forward-Delegate-Receive?)

$$(\nu x\bar{x})(x(y).\,P \parallel \bar{x}{\leftrightarrow}w)$$

$$\triangleq$$

$$(\nu x\bar{x})(x[y].\,P \parallel w(z).\,\bar{x}[\bar{y}].\,(\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

WHAT DO? ③ MAKE IT LAZY

(Forward-Delegate)

$$(\nu x\bar{x})(x[y].\, P \parallel \bar{x}{\leftrightarrow}w)$$

$$\triangleq$$

$$(\nu x\bar{x})(x[y].\, P \parallel \bar{x}(\bar{y}).\, w[z].\, (\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

$$\downarrow$$

$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel w[z].\, (\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

(Forward-Delegate-Receive?)

$$(\nu x\bar{x})(x(y).\, P \parallel \bar{x}{\leftrightarrow}w)$$

$$\triangleq$$

$$(\nu x\bar{x})(x[y].\, P \parallel w(z).\, \bar{x}[\bar{y}].\, (\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

UH OH?

WHAT DO? ③ MAKE IT LAZY

This reduces...

$$(\nu x\bar{x})(x[y].\,P \parallel \bar{x}{\leftrightarrow}w)$$

$$\downarrow$$

$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel w[z].\,(\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

...using (Forward-Delegate).

This is stuck.

$$(\nu x\bar{x})(x(y).\,P \parallel \bar{x}{\leftrightarrow}w)$$

Is that bad? No!

(Disclaimer: That's kind of what "forwarder" means, isn't it?)

CONCLUSION: MAKE IT LAZY!

Replace (Forward) with Lazy Identity Expansion...

(Forward-Delegate)

$$(\nu x\bar{x})(x[y].\,P \parallel \bar{x}{\leftrightarrow}w)$$

$$\downarrow$$

$$(\nu x\bar{x})(\nu y\bar{y})(P \parallel w[z].\,(\bar{y}{\leftrightarrow}z \parallel \bar{x}{\leftrightarrow}w))$$

(Forward-Choose)

$$(\nu x\bar{x})(x \triangleleft \mathrm{inl}.\,P \parallel \bar{x}{\leftrightarrow}w)$$

$$\downarrow$$

$$(\nu x\bar{x})(P \parallel w \triangleleft \mathrm{inl}.\,\bar{x}{\leftrightarrow}w)$$

...and the simplified metatheory just works!*

(*: Mostly. The definition of dependency becomes slightly more complicated.)