Towards Concurrent Type Theory

Luís Caires¹, Frank Pfenning², Bernardo Toninho¹,²

¹ Universidade Nova de Lisboa
² Carnegie Mellon University

Workshop on Types in Language Design and Implementation (TLDI)
January 28, 2012
In intuitionistic logic:

- Propositions are simple types
- Proofs are functional programs
- Proof reduction is computation
In intuitionistic logic:
- Propositions are simple types
- Proofs are functional programs
- Proof reduction is computation

Curry (1934)
- Axiomatic proofs are combinators
- Proof reduction is combinatory reduction

Howard (1969)
- Natural deductions are $\lambda$-terms
- Proof reduction is functional computation
In intuitionistic logic:
- Propositions are simple types
- Proofs are functional programs
- Proof reduction is computation

Curry (1934)
- Axiomatic proofs are combinators
- Proof reduction is combinatory reduction

Howard (1969)
- Natural deductions are $\lambda$-terms
- Proof reduction is functional computation

These are isomorphisms!
Other instances

- Capture computational phenomena logically
  - Modal logic (JS4) and staged computation (Davies & Pf. 1996)
  - Temporal logic and partial evaluation (Davies 1996)
  - Lax logic and effects (Benton et al. 1998)
  - Modal logic (JT) and proof irrelevance (Pf. 2008)
  - ... (but not as easy as it looks)
Other instances

- Capture computational phenomena logically
  - Modal logic (JS4) and staged computation (Davies & Pf. 1996)
  - Temporal logic and partial evaluation (Davies 1996)
  - Lax logic and effects (Benton et al. 1998)
  - Modal logic (JT) and proof irrelevance (Pf. 2008)
  - ... (but not as easy as it looks)

- This talk:
  - Linear propositions as session types
  - Sequent proofs as $\pi$-calculus processes
  - Cut reduction as communication
Type theory (Martin-Löf 1980)
- Generalizes intuitionistic logic
- Types depend on programs
- Full integration of reasoning and programming
Type theory

- Type theory (Martin-Löf 1980)
  - Generalizes intuitionistic logic
  - Types depend on programs
  - Full integration of reasoning and programming
- Co-design of language and reasoning principles!
Type theory

- Type theory (Martin-Löf 1980)
  - Generalizes intuitionistic logic
  - Types depend on programs
  - Full integration of reasoning and programming
- Co-design of language and reasoning principles!
- This talk:
  - Session types depend on functional values
  - Communicate channels and values (= proofs)
Type theory

- Type theory (Martin-Löf 1980)
  - Generalizes intuitionistic logic
  - Types depend on programs
  - Full integration of reasoning and programming
- Co-design of language and reasoning principles!
- This talk:
  - Session types depend on functional values
  - Communicate channels and values (= proofs)
- Not yet:
  - Types do not depend on channels or processes
  - Processes are not communicated
Outline

1. Session types for \( \pi \)-calculus
2. Dependent session types
3. Proof irrelevance
4. Some results
5. Conclusion
Judgment forms

- Judgment $P :: x : A$
  - Process $P$ offers service $A$ along channel $x$
- Linear sequent

$$\Delta \Rightarrow A_1, \ldots, A_n \Rightarrow A$$

- Cut as composition

$$\Delta \Rightarrow A \quad \Delta', A \Rightarrow C \quad \text{cut}_A \quad \frac{}{\Delta, \Delta' \Rightarrow C}$$

- Identity as forwarding

$$\text{id}_A \quad \frac{}{A \Rightarrow A}$$
Judgment forms

- Judgment $P :: x : A$
  - Process $P$ offers service $A$ along channel $x$
- Linear sequent

\[
\frac{x_1 : A_1, \ldots, x_n : A_n}{\Delta} \Rightarrow P :: x : A
\]

$P$ uses $x_i : A_i$ and offers $x : A$.

- Cut as composition

\[
\frac{\Delta \Rightarrow A}{\Delta, \Delta' \Rightarrow C} \quad \text{cut}_A
\]

- Identity as forwarding

\[
\frac{A \Rightarrow A}{\text{id}_A}
\]
Judgment forms

- Judgment $P :: x : A$
  - Process $P$ offers service $A$ along channel $x$
- Linear sequent
  \[
 \frac{x_1:A_1, \ldots, x_n:A_n}{P :: x : A} \\
  \Delta
  \]
  $P$ uses $x_i:A_i$ and offers $x:A$.
- Cut as composition
  \[
  \frac{\Delta \Rightarrow x : A \quad \Delta', x:A \Rightarrow z : C}{\Delta, \Delta' \Rightarrow \quad \begin{array}{c} \text{cut}_A \end{array} \quad z : C}
  \]
- Identity as forwarding
  \[
  \frac{\Delta \Rightarrow \quad \text{id}_A}{A \Rightarrow A}
  \]
Judgment forms

- Judgment $P :: x : A$
  - Process $P$ offers service $A$ along channel $x$
- Linear sequent
  \[
  \Delta \vdash x_1:A_1, \ldots, x_n:A_n \Rightarrow P :: x : A
  \]
- $P$ uses $x_i:A_i$ and offers $x:A$.
- Cut as composition
  \[
  \Delta \Rightarrow P :: x : A \quad \Delta', x:A \Rightarrow Q :: z : C
  \]
  \[
  \Delta, \Delta' \Rightarrow (\nu x)(P \mid Q) :: z : C
  \]
  \[
  \text{cut}_A
  \]
- Identity as forwarding
  \[
  A \Rightarrow A \quad \text{id}_A
  \]
Judgment forms

- Judgment $P :: x : A$
- Process $P$ offers service $A$ along channel $x$
- Linear sequent

$$
\begin{aligned}
&x_1 : A_1, \ldots, x_n : A_n \Rightarrow P :: x : A \\
&\Delta
\end{aligned}
$$

$P$ uses $x_i : A_i$ and offers $x : A$.

- Cut as composition

$$
\begin{aligned}
&\Delta \Rightarrow P :: x : A \\
&\Delta', x : A \Rightarrow Q :: z : C
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
&\Delta, \Delta' \Rightarrow (\nu x)(P \mid Q) :: z : C \\
&\text{cut}_A
\end{aligned}
$$

- Identity as forwarding

$$
\begin{aligned}
&x : A \Rightarrow \\
&z : A
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
&z : A
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
&\text{id}_A
\end{aligned}$$
Judgment forms

- **Judgment** $P :: x : A$
  - Process $P$ offers service $A$ along channel $x$
- **Linear sequent**
  
  $$\frac{x_1 : A_1, \ldots, x_n : A_n \Rightarrow P :: x : A}{\Delta}$$

  $P$ uses $x_i : A_i$ and offers $x : A$.

- **Cut as composition**
  
  $$\Delta \Rightarrow P :: x : A \quad \Delta', x : A \Rightarrow Q :: z : C \Rightarrow \frac{\Delta, \Delta' \Rightarrow (\nu x)(P \mid Q) :: z : C}{\text{cut}_A}$$

- **Identity as forwarding**
  
  $$x : A \Rightarrow [x \leftrightarrow z] :: z : A \Rightarrow \text{id}_A$$
Offering input \((A \leadsto B)\)

- \(P :: x : A \leadsto B\)
  - \(P\) inputs an \(A\) along \(x\) and then behaves as \(B\)
- Right rule: offer of service
  \[
  \frac{\Delta, A \Rightarrow B}{\Delta \Rightarrow A \leadsto B} \quad \leadsto_R
  \]
- Left rule: matching use of service
  \[
  \frac{\Delta \Rightarrow A \quad \Delta', B \Rightarrow C}{\Delta, \Delta', A \leadsto B \Rightarrow C} \quad \leadsto_L
  \]
Offering input \((A \rightarrow B)\)

- \(P :: x : A \rightarrow B\)
  - \(P\) inputs an \(A\) along \(x\) and then behaves as \(B\)
- Right rule: offer of service

\[
\frac{\Delta, A \Rightarrow B}{\Delta \Rightarrow A \rightarrow B} \quad \rightarrow \circ R
\]

- Left rule: matching use of service

\[
\frac{\Delta \Rightarrow A \quad \Delta', B \Rightarrow C}{\Delta, \Delta', A \rightarrow B \Rightarrow C} \quad \leftarrow \circ L
\]
Offering input \((A \xrightarrow{\circ} B)\)

- \(P :: x : A \xrightarrow{\circ} B\)
  - \(P\) inputs an \(A\) along \(x\) and then behaves as \(B\)
- Right rule: offer of service
  \[
  \begin{align*}
  \Delta, y:A & \Rightarrow x : B \\
  \Delta & \Rightarrow x : A \xrightarrow{\circ} B
  \end{align*}
  \xrightarrow{\circ} R
  \]
- Left rule: matching use of service
  \[
  \begin{align*}
  \Delta & \Rightarrow A \\
  \Delta', B & \Rightarrow C
  \end{align*}
  \xrightarrow{\circ} L
  \]
  
  Can reuse \(x\), due to linearity!

Channel \(y\) must be new (bound output).
Offering input \((A \rightarrow B)\)

- \(P :: x : A \rightarrow B\)
  - \(P\) inputs an \(A\) along \(x\) and then behaves as \(B\)
- Right rule: offer of service
  \[
  \frac{\Delta, y:A \Rightarrow P :: x : B}{\Delta \Rightarrow x(y).P :: x : A \rightarrow B} \quad \rightarrow R
  \]
  - Can reuse \(x\), due to linearity!
- Left rule: matching use of service
  \[
  \frac{\Delta \Rightarrow A \quad \Delta', B \Rightarrow C}{\Delta, \Delta', A \rightarrow B \Rightarrow C} \quad \leftarrow L
  \]
Offering input \((A \xrightarrow{\circ} B)\)

- \(P :: x : A \xrightarrow{\circ} B\)
  - \(P\) inputs an \(A\) along \(x\) and then behaves as \(B\)
  - Right rule: offer of service

\[
\begin{align*}
\Delta, y: A &\Rightarrow P :: x : B \\
\Delta &\Rightarrow x(y).P :: x : A \xrightarrow{\circ} B \quad \xrightarrow{\circ} R
\end{align*}
\]

- Can reuse \(x\), due to linearity!

- Left rule: matching use of service

\[
\begin{align*}
\Delta &\Rightarrow A & \Delta', B &\Rightarrow C \\
\Delta, \Delta', A \xrightarrow{\circ} B &\Rightarrow C \quad \xrightarrow{\circ} L
\end{align*}
\]
Offering input \((A \rightarrow B)\)

- \(P :: x : A \rightarrow B\)
  - \(P\) inputs an \(A\) along \(x\) and then behaves as \(B\)
- Right rule: offer of service

\[
\frac{\Delta, y:A \Rightarrow P :: x:B}{\Delta \Rightarrow x(y).P :: x:A \rightarrow B}
\]

- Can reuse \(x\), due to linearity!
- Left rule: matching use of service

\[
\frac{\Delta \Rightarrow y : A \quad \Delta', x:B \Rightarrow z : C}{\Delta, \Delta', x:A \rightarrow B \Rightarrow \Delta, \Delta', z : C}
\]
Offering input \((A \rightarrow B)\)

\[ P :: x : A \rightarrow B \]

- \(P\) inputs an \(A\) along \(x\) and then behaves as \(B\)

Right rule: offer of service

\[
\Delta, y : A \Rightarrow P :: x : B \\
\Delta \Rightarrow x(y).P :: x : A \rightarrow B \quad \rightarrow\!\!\!\!\!\!\!\!R
\]

- Can reuse \(x\), due to linearity!

Left rule: matching use of service

\[
\Delta \Rightarrow P :: y : A \quad \Delta', x : B \Rightarrow Q :: z : C \\
\Delta, \Delta', x : A \rightarrow B \Rightarrow (\nu y)x(y).P \mid Q :: z : C \quad \rightarrow\!\!\!\!\!\!\!\!L
\]

- Can reuse \(x\), due to linearity
- Channel \(y\) must be new (bound output)
Proof and process reduction

- **Proof reduction**

  \[
  \Delta, A \Rightarrow B \quad \Delta_1 \Rightarrow A \quad \Delta_2, B \Rightarrow C
  \]

  \[
  \Delta \Rightarrow A \rightarrow B \quad \Delta_1, \Delta_2, A \rightarrow B \Rightarrow C
  \]

  \[
  \Delta, A \Rightarrow B \quad \Delta_1 \Rightarrow A \quad \Delta_2, B \Rightarrow C
  \]

  \[
  \Delta, \Delta_1, \Delta_2 \Rightarrow C
  \]

- **Corresponding process reduction**

  \[
  \Delta, \Delta_1, \Delta_2 \Rightarrow (\nu x)(x(y).P_1 \mid (\nu w)(\overline{x}(w).(P_2 \mid Q))) :: z : C
  \]

  \[
  \Delta, \Delta_1, \Delta_2 \Rightarrow (\nu x)((\nu w)(P_2 \mid P_1 \{ w/y \}) \mid Q) :: z : C
  \]
Corresponding process reduction

\[(\nu x)(x(y).P_1 | (\nu w)(\overline{x}\langle w \rangle.(P_2 | Q)))) \rightarrow (\nu x)((\nu w)(P_2 | P_1\{w/y\}) | Q)\]

Instance of (modulo structural congruence)

\[(x(y).P | \overline{x}\langle w \rangle.Q) \rightarrow (P\{w/y\} | Q)\]

Synchronous \(\pi\)-calculus

Typing modulo structural congruence
Linear propositions as session types

- $P :: x : A \rightarrow B$: Input a $y:A$ along $x$ and behave as $B$
- $P :: x : A \otimes B$: Output new $y:A$ along $x$ and behave as $B$
- $P :: x : 1$: Terminate session on $x$
- $P :: x : A \& B$: Offer choice between $A$ and $B$ along $x$
- $P :: x : A \oplus B$: Offer either $A$ or $B$ along $x$
- $P :: x : !A$: Offer $A$ persistently along $x$

Sequent proofs as process expressions

Proof reduction as process reduction
Offering output \((A \otimes B)\)

- \(P :: x : A \otimes B\)
  - \(P\) outputs a fresh \(y : A\) along \(x\) and then behaves as \(B\)
- Right rule: offer output
  \[
  \frac{\Delta \Rightarrow A \quad \Delta' \Rightarrow B}{\Delta, \Delta' \Rightarrow A \otimes B} \otimes R
  \]
- Left rule: perform matching input
  \[
  \frac{\Delta, A, B \Rightarrow C}{\Delta, A \otimes B \Rightarrow C} \otimes L
  \]
Offering output \((A \otimes B)\)

- \(P :: x : A \otimes B\)
  - \(P\) outputs a fresh \(y : A\) along \(x\) and then behaves as \(B\)
- Right rule: offer output

\[
\begin{align*}
\Delta & \Rightarrow & A & \Delta' & \Rightarrow & B \\
\Delta, \Delta' & \Rightarrow & A \otimes B & \otimes R
\end{align*}
\]

- Left rule: perform matching input

\[
\begin{align*}
\Delta, A, B & \Rightarrow C \\
\Delta, A \otimes B & \Rightarrow C & \otimes L
\end{align*}
\]
Offering output \((A \otimes B)\)

- \(P :: x : A \otimes B\)
  - \(P\) outputs a fresh \(y : A\) along \(x\) and then behaves as \(B\)

- Right rule: offer output

\[
\begin{align*}
\Delta & \Rightarrow y : A \\
\Delta' & \Rightarrow x : B
\end{align*}
\]

\(\Delta, \Delta' \Rightarrow x : A \otimes B \otimes R\)

- Left rule: perform matching input

\[
\begin{align*}
\Delta, A, B & \Rightarrow C
\end{align*}
\]

\(\Delta, A \otimes B \Rightarrow C \otimes L\)
Offering output \((A \otimes B)\)

- \(P :: x : A \otimes B\)
  - \(P\) outputs a fresh \(y:A\) along \(x\) and then behaves as \(B\)
- Right rule: offer output
  \[
  \begin{align*}
  \Delta & \Rightarrow P :: y : A \\
  \Delta' & \Rightarrow Q :: x : B \\
  \Delta, \Delta' & \Rightarrow (\nu y)\overline{x}\langle y \rangle.(P | Q) :: x : A \otimes B
  \end{align*}
  \]
  \(\otimes R\)

- Left rule: perform matching input
  \[
  \begin{align*}
  \Delta, A, B & \Rightarrow C \\
  \Delta, A \otimes B & \Rightarrow C
  \end{align*}
  \]
  \(\otimes L\)
Offering output \((A \otimes B)\)

- \(P :: x : A \otimes B\)
  - \(P\) outputs a fresh \(y : A\) along \(x\) and then behaves as \(B\)

- Right rule: offer output

\[
\frac{\Delta \Rightarrow P :: y : A \quad \Delta' \Rightarrow Q :: x : B}{\Delta, \Delta' \Rightarrow (\nu y) \overline{x} \langle y \rangle . (P \mid Q) :: x : A \otimes B}
\]

- Left rule: perform matching input

\[
\frac{\Delta, A, B \Rightarrow C}{\Delta, A \otimes B \Rightarrow C}
\]

\(\otimes R\)

\(\otimes L\)
Offering output \((A \otimes B)\)

- \(P :: x : A \otimes B\)
  - \(P\) outputs a fresh \(y:A\) along \(x\) and then behaves as \(B\)
- Right rule: offer output

\[
\Delta \Rightarrow P :: y : A \quad \Delta' \Rightarrow Q :: x : B \\
\Delta, \Delta' \Rightarrow (\nu y)\overline{x}\langle y\rangle.(P | Q) :: x : A \otimes B
\]

- Left rule: perform matching input

\[
\Delta, y:A, x:B \Rightarrow z : C \\
\Delta, x:A \otimes B \Rightarrow z : C
\]
Offering output \((A \otimes B)\)

- \(P :: x : A \otimes B\)
  - \(P\) outputs a fresh \(y : A\) along \(x\) and then behaves as \(B\)
- Right rule: offer output

\[
\Delta \Rightarrow P :: y : A \quad \Delta' \Rightarrow Q :: x : B
\]

\[
\Delta, \Delta' \Rightarrow (\nu y)x\langle y \rangle.(P | Q) :: x : A \otimes B
\]

- Left rule: perform matching input

\[
\Delta, y:A, x:B \Rightarrow P :: z : C
\]

\[
\Delta, x:A \otimes B \Rightarrow x(y).P :: z : C
\]
Offering output \((A \otimes B)\)

- Proof reduction again corresponds to process reduction
- No new rules required
- Apparent asymmetry, but \(A \otimes B \simeq B \otimes A\):

\[
x : A \otimes B \Rightarrow x(y).(νw)z\langle w \rangle.([x \leftrightarrow w] \mid [y \leftrightarrow z]) :: z : B \otimes A
\]
Termination (1)

- $P :: x : 1$
  - $P$ terminates session on $x$
- Right rule: offer of termination (unit of $\otimes$)
  \[
  \Delta \Rightarrow 1 \quad \Rightarrow 1R
  \]
- Left rule: accept termination
  \[
  \Delta \Rightarrow C \quad \Rightarrow \Delta, 1 \Rightarrow C \quad 1L
  \]
- Reduction
Termination (1)

- $P :: x : 1$
  - $P$ terminates session on $x$
- Right rule: offer of termination (unit of $\otimes$)
  
  \[
  \begin{array}{c}
  \Delta \Rightarrow C \\
  \hline
  \Delta, 1 \Rightarrow C
  \end{array}
  \]
  
  $1R$

- Left rule: accept termination

  \[
  \begin{array}{c}
  \Delta \Rightarrow C \\
  \hline
  \Delta, 1 \Rightarrow C
  \end{array}
  \]

  $1L$

- Reduction
Termination (1)

- $P :: x : 1$
  - $P$ terminates session on $x$
- Right rule: offer of termination (unit of $\otimes$)
  \[
  \frac{}{\Rightarrow x : 1} \text{1R}
  \]
- Left rule: accept termination
  \[
  \frac{\Delta \Rightarrow C}{\Delta, 1 \Rightarrow C} \text{1L}
  \]
- Reduction
Termination (1)

- $P :: x : 1$
  - $P$ terminates session on $x$
- Right rule: offer of termination (unit of $\otimes$)

\[
\begin{align*}
\text{Right rule: offer of termination (unit of } \otimes) \\
\hline
\Rightarrow \bar{x}\langle \rangle \cdot 0 :: x : 1 \quad 1^R
\end{align*}
\]
- Left rule: accept termination

\[
\begin{align*}
\Delta \Rightarrow C \\
\hline
\Delta, 1 \Rightarrow C \quad 1^L
\end{align*}
\]

- Reduction
Termination (1)

- $P :: x : 1$
  - $P$ terminates session on $x$

- Right rule: offer of termination (unit of $\otimes$)

  $\quad \Rightarrow \bar{x}\langle \rangle .0 :: x : 1$

- Left rule: accept termination

  $\quad \Delta \Rightarrow C$

  $\Delta, 1 \Rightarrow C$

- Reduction
Termination (1)

- $P : x : 1$
  - $P$ terminates session on $x$
- Right rule: offer of termination (unit of $\otimes$)
  \[
  \frac{}{\Rightarrow x\langle\rangle.0 : x : 1} \quad 1R
  \]
- Left rule: accept termination
  \[
  \frac{\Delta \Rightarrow z : C}{\Delta, x:1 \Rightarrow z : C} \quad 1L
  \]
- Reduction
Termination (1)

- $P :: x : 1$
  - $P$ terminates session on $x$

  **Right rule: offer of termination (unit of $\otimes$)**

  $\Rightarrow \bar{x}(\cdot) . 0 :: x : 1$  \hspace{1cm} $1^R$

- Left rule: accept termination

  $\Delta \Rightarrow P :: z : C$

  $\Delta, x : 1 \Rightarrow x().P :: z : C$  \hspace{1cm} $1^L$

- Reduction
Termination (1)

- \( P :: x : 1 \)
  - \( P \) terminates session on \( x \)
- Right rule: offer of termination (unit of \( \otimes \))
  \[
  \frac{\text{\( \Rightarrow \)} \quad \overline{x}() \cdot 0 :: x : 1}{\text{\( \Rightarrow \)} \quad \overline{x}() \cdot 0 :: x : 1} \quad 1R
  \]
- Left rule: accept termination
  \[
  \frac{\Delta \Rightarrow P :: z : C}{\Delta, x : 1 \Rightarrow x().P :: z : C} \quad 1L
  \]
- Reduction
  \[
  (\overline{x}() \cdot 0 \mid x().P) \rightarrow P
  \]
Termination (1)

- This faithful process assignment models synchronous termination
- We can also model asynchronous termination
  - Use a different process assignment (Caires & Pf. 2010)
  - Contracting proofs to processes
  - Some proof reductions are process identities
Example: PDF indexing

- Abstract away communicated values for now

\[ \text{index}_1 \triangleq \text{file} \rightarrow (\text{file} \otimes 1) \]

- Shape of a server

\[ \text{srv} \triangleq x(f). (\nu y) x(y).(P | y).0 :: x : \text{index}_1 \]

- Shape of a client

\[ \text{client} \triangleq (\nu pdf) x(pdf). x(idx). x().Q \]

- Composition of server and client

\[ \vdash \text{srv} :: x : \text{index}_1 \quad x : \text{index}_1 \Rightarrow \text{client} :: z : 1 \]

\[ \quad \frac{}{\vdash (\nu x)(\text{srv} | \text{client}) :: z : 1} \text{ cut} \]
At this point we have

**Types**  \( A, B, C \quad ::= \quad A \rightarrow B \) input
\[ A \otimes B \] output
\[ 1 \] termination

**Processes**  \( P, Q \quad ::= \quad [x \leftrightarrow z] \) forwarding
\[ (P | Q) \] parallel composition
\[ (\nu x)P \] name restriction
\[ x(y).P \] input
\[ (\nu y)\bar{x}\langle y \rangle .P \] bound output
\[ x().P \] (wait)
\[ \bar{x}\langle \rangle .0 \] (termination)
Offering external choice ($A \& B$)

- $P :: x : A \& B$
  - $P$ offers the choice between $A$ and $B$ along $x$
- Right rule: offering choice between $A$ and $B$
  $$\d\Rightarrow A \quad \d\Rightarrow B \quad \&R
  \d\Rightarrow A \& B$$

- Left rules: making a choice between $A$ and $B$
  $$\d, A \Rightarrow C \quad \&L_1
  \d, A \& B \Rightarrow C$$
  $$\d, B \Rightarrow C \quad \&L_2
  \d, A \& B \Rightarrow C$$
Offering external choice \((A \& B)\)

- \(P :: x : A \& B\)
  - \(P\) offers the choice between \(A\) and \(B\) along \(x\)
- Right rule: offering choice between \(A\) and \(B\)
  
  \[
  \Delta \Rightarrow \quad A \quad \Delta \Rightarrow \quad B
  \]
  
  \[
  \Delta \Rightarrow \quad A \& B
  \]
  
  \(\&R\)

- Left rules: making a choice between \(A\) and \(B\)
  
  \[
  \Delta, A \Rightarrow C
  \]
  
  \[
  \Delta, A \& B \Rightarrow C \quad \&L_1
  \]

  \[
  \Delta, B \Rightarrow C
  \]
  
  \[
  \Delta, A \& B \Rightarrow C \quad \&L_2
  \]
Offering external choice \((A \& B)\)

- \(P :: x : A \& B\)
  - \(P\) offers the choice between \(A\) and \(B\) along \(x\)
- Right rule: offering choice between \(A\) and \(B\)
  \[
  \Delta \Rightarrow x : A \quad \Delta \Rightarrow x : B \\
  \Delta \Rightarrow x : A \& B \quad \&R
  \]
- Left rules: making a choice between \(A\) and \(B\)
  \[
  \Delta, A \Rightarrow C \\
  \Delta, A \Rightarrow C \\
  \Delta, B \Rightarrow C \\
  \Delta, A \& B \Rightarrow C \quad \&L_1
  \]
  \[
  \Delta, A \& B \Rightarrow C \\
  \Delta, A \& B \Rightarrow C \quad \&L_2
  \]
Offering external choice \((A \& B)\)

- \(P :: x : A \& B\)
  - \(P\) offers the choice between \(A\) and \(B\) along \(x\)
- Right rule: offering choice between \(A\) and \(B\)

\[
\Delta \Rightarrow P :: x : A \quad \Delta \Rightarrow Q :: x : B
\]

\[
\Delta \Rightarrow x.\text{case}(P, Q) :: x : A \& B \quad &R
\]

- Left rules: making a choice between \(A\) and \(B\)

\[
\Delta, A \Rightarrow C \quad \Delta, A \& B \Rightarrow C \quad &L_1
\]

\[
\Delta, B \Rightarrow C \quad \Delta, A \& B \Rightarrow C \quad &L_2
\]
Offering external choice (A & B)

- \( P :: x : A \& B \)
- \( P \) offers the choice between \( A \) and \( B \) along \( x \)
- Right rule: offering choice between \( A \) and \( B \)

\[
\begin{align*}
\Delta & \Rightarrow P :: x : A \\
\Delta & \Rightarrow Q :: x : B \\
\Delta & \Rightarrow x.\text{case}(P, Q) :: x : A \& B
\end{align*}
\]

\( \&R \)

- Left rules: making a choice between \( A \) and \( B \)

\[
\begin{align*}
\Delta, & \quad A \Rightarrow C & \Delta \Rightarrow C & \&L_1 \\
\Delta, & \quad A \& B \Rightarrow C \\
\Delta, & \quad B \Rightarrow C & \Delta \Rightarrow C & \&L_2 \\
\Delta, & \quad A \& B \Rightarrow C
\end{align*}
\]
Offering external choice \((A \& B)\)

- \(P :: x : A \& B\)
  - \(P\) offers the choice between \(A\) and \(B\) along \(x\)
- Right rule: offering choice between \(A\) and \(B\)

\[
\Delta \Rightarrow P :: x : A \quad \Delta \Rightarrow Q :: x : B
\]
\[
\Delta \Rightarrow x.\text{case}(P, Q) :: x : A \& B
\]

- Left rules: making a choice between \(A\) and \(B\)

\[
\Delta, x:A \Rightarrow z : C \quad \Delta, x:A & B \Rightarrow z : C \quad &L_1
\]
\[
\Delta, x:B \Rightarrow z : C \quad \Delta, x:A & B \Rightarrow z : C \quad &L_2
\]
Offering external choice \((A \& B)\)

- **\(P :: x : A \& B\)**
  - \(P\) offers the choice between \(A\) and \(B\) along \(x\)
- **Right rule: offering choice between \(A\) and \(B\)**
  
  \[
  \Delta \Rightarrow P :: x : A \quad \Delta \Rightarrow Q :: x : B \\
  \Delta \Rightarrow x.\text{case}(P, Q) :: x : A \& B
  \]

  &R

- **Left rules: making a choice between \(A\) and \(B\)**

  \[
  \Delta, x:A \Rightarrow Q :: z : C \\
  \Delta, x:A \& B \Rightarrow x.\text{inl}; Q :: z : C \quad \&L_1
  \]

  \[
  \Delta, x:B \Rightarrow Q :: z : C \\
  \Delta, x:A \& B \Rightarrow x.\text{inr}; Q :: z : C \quad \&L_2
  \]
Offering external choice ($A \& B$)

- Need binary guarded choice construct
- New reductions

\[
(x.\text{case}(P, Q) \mid x.\text{inl}; R) \rightarrow (P \mid R)
\]
\[
(x.\text{case}(P, Q) \mid x.\text{inr}; R) \rightarrow (Q \mid R)
\]
Example: PDF compression

- Extend previous example
- Offer to index or compress the PDF

\[
\text{server}_1 \triangleq (\text{file} \rightarrow (\text{file} \otimes 1)) \\
\quad \land (\text{file} \rightarrow (\text{file} \otimes 1))
\]

- Different protocol: decision is made later

\[
\text{server}_2 \triangleq \text{file} \rightarrow ((\text{file} \land \text{file}) \otimes 1)
\]

- In practice, should use labeled products \(\& \{l_i : A_i\}\)
Offering internal choice \((A \oplus B)\)

- \(P :: x : A \oplus B\)
  - \(P\) offers either \(A\) or \(B\) along \(x\)
  - Offering either \(A\) or \(B\):

\[
\begin{align*}
\Delta \Rightarrow A & \quad \oplus R_1 \\
\Delta \Rightarrow A \oplus B & \\
\Delta \Rightarrow B & \quad \oplus R_2 \\
\Delta \Rightarrow A \oplus B & \\
\end{align*}
\]

- Accounting for either \(A\) or \(B\):

\[
\begin{align*}
\Delta, A \Rightarrow C & \quad \Delta, B \Rightarrow C \\
\Delta, A \oplus B \Rightarrow C & \quad \oplus L \\
\end{align*}
\]

- No new reductions
Offering internal choice \((A \oplus B)\)

- **\(P :: x : A \oplus B\)**
  - \(P\) offers either \(A\) or \(B\) along \(x\)

- Offering either \(A\) or \(B\):
  
  \[
  \frac{\Delta \Rightarrow A}{\Delta \Rightarrow A \oplus B} \quad \oplus R_1
  \]
  \[
  \frac{\Delta \Rightarrow B}{\Delta \Rightarrow A \oplus B} \quad \oplus R_2
  \]

- Accounting for either \(A\) or \(B\):
  
  \[
  \frac{\Delta, A \Rightarrow C \quad \Delta, B \Rightarrow C}{\Delta, A \oplus B \Rightarrow C} \quad \oplus L
  \]

- No new reductions
Offering internal choice \((A \oplus B)\)

- \(P :: x : A \oplus B\)
  - \(P\) offers either \(A\) or \(B\) along \(x\)
- Offering either \(A\) or \(B\):
  
  \[
  \Delta \Rightarrow \quad x : A \quad \oplus R_1
  \]
  \[
  \Delta \Rightarrow \quad x : A \oplus B
  \]
  
  \[
  \Delta \Rightarrow \quad x : B \quad \oplus R_2
  \]
  \[
  \Delta \Rightarrow \quad x : A \oplus B
  \]

- Accounting for either \(A\) or \(B\):
  
  \[
  \Delta, A \Rightarrow C \quad \Delta, B \Rightarrow C \quad \oplus L
  \]
  \[
  \Delta, A \oplus B \Rightarrow C
  \]

- No new reductions
Offering internal choice \((A \oplus B)\)

- \(P :: x : A \oplus B\)
  - \(P\) offers either \(A\) or \(B\) along \(x\)
- Offering either \(A\) or \(B\):
  \[
  \begin{align*}
  \Delta \Rightarrow P :: x : A & \quad \oplus R_1 \\
  \Delta \Rightarrow x.\text{inl}; P :: x : A \oplus B \\
  \Delta \Rightarrow P :: x : B & \quad \oplus R_2 \\
  \Delta \Rightarrow x.\text{inr}; P :: x : A \oplus B
  \end{align*}
  \]
- Accounting for either \(A\) or \(B\):
  \[
  \begin{align*}
  \Delta, A \Rightarrow C & \quad \Delta, B \Rightarrow C \\
  \Delta, A \oplus B \Rightarrow C & \quad \oplus L
  \end{align*}
  \]
- No new reductions
Offering internal choice \((A \oplus B)\)

- \(P :: x : A \oplus B\)
  - \(P\) offers either \(A\) or \(B\) along \(x\)
- Offering either \(A\) or \(B\):
  \[
  \begin{align*}
  \Delta \Rightarrow P :: x : A & \quad \oplus R_1 \\
  \Delta \Rightarrow x.\text{inl}; P :: x : A \oplus B & \\
  \Delta \Rightarrow P :: x : B & \quad \oplus R_2 \\
  \Delta \Rightarrow x.\text{inr}; P :: x : A \oplus B & 
  \end{align*}
  \]
- Accounting for either \(A\) or \(B\):
  \[
  \begin{align*}
  \Delta, \quad A \Rightarrow & \quad C & \Delta, \quad B \Rightarrow & \quad C & \quad \oplus L \\
  \Delta, \quad A \oplus B \Rightarrow & \quad C
  \end{align*}
  \]
- No new reductions
Offering internal choice \((A \oplus B)\)

- \(P :: x : A \oplus B\)
  - \(P\) offers either \(A\) or \(B\) along \(x\)
  - Offering either \(A\) or \(B\):

\[
\Delta \Rightarrow P :: x : A \\
\Delta \Rightarrow x.\text{inl}; P :: x : A \oplus B \quad \oplus R_1
\]

\[
\Delta \Rightarrow P :: x : B \\
\Delta \Rightarrow x.\text{inr}; P :: x : A \oplus B \quad \oplus R_2
\]

- Accounting for either \(A\) or \(B\):

\[
\Delta, x:A \Rightarrow z : C \\
\Delta, x:B \Rightarrow z : C \quad \oplus L
\]

- No new reductions
Offering internal choice \((A \oplus B)\)

- \(P :: x : A \oplus B\)
  - \(P\) offers either \(A\) or \(B\) along \(x\)

Offering either \(A\) or \(B\):

\[
\begin{align*}
\Delta \Rightarrow P :: x : A & \quad \oplus R_1 \\
\Delta \Rightarrow x.\text{inl}; P :: x : A \oplus B & \\
\Delta \Rightarrow P :: x : B & \quad \oplus R_2 \\
\Delta \Rightarrow x.\text{inr}; P :: x : A \oplus B &
\end{align*}
\]

Accounting for either \(A\) or \(B\):

\[
\begin{align*}
\Delta, x:A \Rightarrow P :: z : C \\
\Delta, x:B \Rightarrow Q :: z : C & \quad \oplus L \\
\Delta, x:A \oplus B \Rightarrow x.\text{case}(P, Q) :: z : C
\end{align*}
\]

- No new reductions
Example: PDF indexing

- Offer to index PDF, or indicate failure

\[
\text{index}_2 \triangleq (\text{file} \circ ((\text{file} \otimes 1) \oplus 1))
\]

- In practice, should use labeled sums \( \bigoplus_i \{ l_i : A_i \} \)
To have persistent services, we generalize the judgment form (Hodas & Miller 1991; Andreoli 1992; Barber 1996)

\[ B_1, \ldots, B_k \quad ; \quad A_1, \ldots, A_n \quad \Rightarrow \quad C \]

\[ \Gamma \quad \Delta \]

persistently true \hspace{2cm} linearly true

Label with shared channels \textit{u} and linear channels \textit{x}

\[ u_1:B_1, \ldots, u_k:B_k \quad ; \quad x_1:A_1, \ldots, x_n:A_n \quad \Rightarrow \quad P :: z : C \]

\[ \Gamma \quad \Delta \]

shared \hspace{2cm} linear
Structural rules

- cut! as composition with replicated input

\[
\frac{\Gamma ; \cdot \Rightarrow A \quad \Gamma, A ; \Delta \Rightarrow C}{\Gamma ; \Delta \Rightarrow C} \text{ cut}_A!
\]

- No linear channels in \( P \) except \( x \)
- To use \( u \) we have to send it a new channel \( y \) for \( x \)

\[
\frac{\Gamma, A ; \Delta, A \Rightarrow C}{\Gamma, A ; \Delta \Rightarrow C} \text{ copy}
\]


- **cut!** as composition with replicated input

\[
\begin{align*}
\Gamma ; \cdot & \Rightarrow A \\
\Gamma & \Rightarrow \!
\end{align*}
\]

- No linear channels in \( P \) except \( x \)

- To use \( u \) we have to send it a new channel \( y \) for \( x \)

\[
\begin{align*}
\Gamma, A & ; \Delta, A \Rightarrow C \\
\Gamma & \Rightarrow C
\end{align*}
\]
Structural rules

- cut! as composition with replicated input

\[
\frac{\Gamma ; \cdot \Rightarrow x : A \quad \Gamma, u:A ; \Delta \Rightarrow z : C}{\Gamma ; \Delta \Rightarrow \quad z : C} \quad \text{cut}!_A
\]

- No linear channels in \( P \) except \( x \)

- To use \( u \) we have to send it a new channel \( y \) for \( x \)

\[
\frac{\Gamma, A ; \Delta, A \Rightarrow C}{\Gamma, A ; \Delta \Rightarrow C} \quad \text{copy}
\]
Structural rules

- cut! as composition with replicated input

\[
\begin{align*}
\Gamma ; \cdot & \Rightarrow P :: x : A \\
\Gamma, u : A ; \Delta & \Rightarrow Q :: z : C \\
\Gamma ; \Delta & \Rightarrow (\nu u)(!u(x).P \mid Q) :: z : C
\end{align*}
\]

- No linear channels in \( P \) except \( x \)
- To use \( u \) we have to send it a new channel \( y \) for \( x \)

\[
\frac{\Gamma, A ; \Delta, A \Rightarrow C}{\Gamma, A ; \Delta \Rightarrow C}
\]

\text{copy}
Structural rules

- **cut!** as composition with replicated input

\[
\frac{Γ ; · ⇒ P :: x : A \quad Γ, u:A ; Δ ⇒ Q :: z : C}{Γ ; Δ ⇒ (νu)(!u(x).P | Q) :: z : C}
\]

- No linear channels in \(P\) except \(x\)
- To use \(u\) we have to send it a new channel \(y\) for \(x\)

\[
\frac{Γ, A ; Δ, A ⇒ C}{Γ, A ; Δ ⇒ C}
\]

{\text{copy}}
cut! as composition with replicated input

\[
\frac{\Gamma \vdash \cdot \implies P :: x : A \quad \Gamma, u : A ; \Delta \implies Q :: z : C}{\Gamma \vdash \Delta \implies (\nu u)(!u(x).P \mid Q) :: z : C} \quad \text{cut!}_A
\]

- No linear channels in \( P \) except \( x \)
- To use \( u \) we have to send it a new channel \( y \) for \( x \)

\[
\frac{\Gamma, u : A ; \Delta, y : A \implies z : C}{\Gamma, u : A ; \Delta \implies z : C} \quad \text{copy}
\]
Structural rules

- **cut!** as composition with replicated input

\[
\frac{\Gamma ; \cdot \Rightarrow P :: x : A \quad \Gamma, u:A ; \Delta \Rightarrow Q :: z : C}{\Gamma ; \Delta \Rightarrow (\nu u)(!u(x).P | Q) :: z : C} \quad \text{cut!}_A
\]

- No linear channels in \( P \) except \( x \)

- To use \( u \) we have to send it a new channel \( y \) for \( x \)

\[
\frac{\Gamma, u:A ; \Delta, y:A \Rightarrow P :: z : C}{\Gamma, u:A ; \Delta \Rightarrow (\nu y)\bar{u}\langle y\rangle.P :: z : C} \quad \text{copy}
\]

- \( y \) will be linear and behave according to \( A \)
Replaying the proof reduction yields:

$$(\nu u) (!u(x).P \mid (\nu y) \overline{u}\langle y\rangle .Q)$$

$$\longrightarrow (\nu y) (P\{y/x\} \mid (\nu u) (!u(x).P \mid Q))$$

Instance of standard rule

$$(!u(x).P \mid \overline{u}\langle y\rangle .Q) \longrightarrow (P\{y/x\} \mid Q \mid !u(x).P)$$
Offering persistent service (!A)

- Internalize persistence as a proposition
  - $P :: x :: !A$
    - $P$ persistently offers $A$ along $x$
- Creating a persistent service

$$
\frac{\Gamma ; \cdot \Rightarrow A}{\Gamma ; \cdot \Rightarrow !A} \quad !R
$$

- Sharing a persistent service

$$
\frac{\Gamma, A ; \Delta \Rightarrow C}{\Gamma ; \Delta, !A \Rightarrow C} \quad !L
$$
Offering persistent service (!A)

- Internalize persistence as a proposition
- \( P :: x : !A \)
  - \( P \) persistently offers \( A \) along \( x \)
- Creating a persistent service

\[
\begin{align*}
\Gamma ; \cdot & \Rightarrow A \\
\Gamma & \Rightarrow !A \\
\Gamma ; \cdot & \Rightarrow !A
\end{align*}
\]

- Sharing a persistent service

\[
\begin{align*}
\Gamma, A ; \Delta & \Rightarrow C \\
\Gamma & \Rightarrow \Delta, !A \Rightarrow C
\end{align*}
\]
Offering persistent service (!A)

- Internalize persistence as a proposition
- \( P :: x : !A \)
  - \( P \) persistently offers \( A \) along \( x \)
- Creating a persistent service
  \[
  \frac{\Gamma ; \cdot \Rightarrow y : A}{\Gamma ; \cdot \Rightarrow x : !A} \quad !R
  \]
- Sharing a persistent service
  \[
  \frac{\Gamma, A ; \Delta \Rightarrow C}{\Gamma ; \Delta, !A \Rightarrow C} \quad !L
  \]
Offering persistent service (!A)

- Internalize persistence as a proposition
- \( P :: x : !A \)
  - \( P \) persistently offers \( A \) along \( x \)
- Creating a persistent service
  \[
  \Gamma ; \cdot \Rightarrow P :: y : A
  \]
  \[
  \frac{!R}{\Gamma ; \cdot \Rightarrow !x(y).P :: x : !A}
  \]
- Sharing a persistent service
  \[
  \Gamma, A ; \Delta \Rightarrow C
  \]
  \[
  \frac{!L}{\Gamma ; \Delta, !A \Rightarrow C}
  \]
Offering persistent service (!A)

- Internalize persistence as a proposition
  \[ P :: x : !A \]
  - \( P \) persistently offers \( A \) along \( x \)

- Creating a persistent service
  \[
  \Gamma ; \cdot \Rightarrow P :: y : A
  \]
  \[
  \Gamma ; \cdot \Rightarrow !x(y).P :: x : !A
  \]

- Sharing a persistent service
  \[
  \Gamma, A ; \Delta \Rightarrow C
  \]
  \[
  \Gamma ; \Delta, !A \Rightarrow C
  \]
  \[
  !L
  \]
Offering persistent service (!A)

- Internalize persistence as a proposition
- \( P :: x : !A \)
  - \( P \) persistently offers \( A \) along \( x \)
- Creating a persistent service
  \[
  \Gamma ; \cdot \Rightarrow P :: y : A \\
  \Gamma ; \cdot \Rightarrow !x(y).P :: x : !A
  \]
- Sharing a persistent service
  \[
  \Gamma, u:A ; \Delta \Rightarrow z : C \\
  \Gamma ; \Delta, x!:A \Rightarrow z : C
  \]

\( !R \)

\( !L \)
Offering persistent service (!A)

- Internalize persistence as a proposition
  - \( P :: x : !A \)
    - \( P \) persistently offers \( A \) along \( x \)
- Creating a persistent service
  \[
  \Gamma ; \cdot \Rightarrow P :: y : A \\
  \frac{\Gamma ; \cdot \Rightarrow !x(y).P :: x : !A}{} !R
  \]
- Sharing a persistent service
  \[
  \Gamma, u:A ; \Delta \Rightarrow Q :: z : C \\
  \frac{\Gamma ; \Delta, x:!A \Rightarrow x/u.Q :: z : C}{} !L
  \]
Promotion

- !L promotes linear channels to shared ones
- No significant operational consequences

\[ (\nu x)(!x(y).P \mid x/u.Q) \rightarrow (\nu u)(!u(y).P \mid Q) \]
Example: persistent storage

- Persistent PDF indexing service

\[
\text{index}_3 : ! (\text{file} \to \text{file} \otimes 1)
\]

Persistently offer to input a file, then output a file and terminate session.

- Store a file persistently

\[
\text{store}_1 : ! (\text{file} \to ! (\text{file} \otimes 1))
\]

Persistently offer to input a file, then output a persistent handle for retrieving this file.
At this point we have in addition

**Types**
\[ A, B, C ::= \ldots \]

- \( A \& B \) external choice
- \( A \oplus B \) internal choice
- \(!A\) replication

**Processes**
\[ P, Q ::= \ldots \]

- \( x.\text{inl}; P \mid x.\text{inr}; P \) selection
- \( x.\text{case}(P, Q) \) branching
- \( !u(x).P \) replicating input
- \( x/u.P \) (promotion)
Outline

1. Session types for \( \pi \)-calculus
2. Dependent session types
3. Proof irrelevance
4. Some results
5. Conclusion
Passing terms

- Types $\tau$ from a (dependent) type theory
- Hypothetical judgment $\vdash M : \tau$

Some example type constructors

- $\Pi x : \tau. \sigma$, $\tau \rightarrow \sigma$  Functions from $\tau$ to $\sigma$
- $\Sigma x : \tau. \sigma$, $\tau \times \sigma$  Pairs of a $\tau$ and a $\sigma$
- nat  Natural numbers

Integrate into sequent calculus

$\psi ; \Gamma ; \Delta \Rightarrow P :: x : A$

- $\psi$ term variables
- $\Gamma$ shared channels
- $\Delta$ linear channels
- $A$ linear
Offering term input ($\forall y: \tau. A$)

- $P :: x : \forall y : \tau. A$
  - $P$ inputs an $M : \tau$ along $x$ and then behaves as $A\{M/x\}$

- Right rule: offer of service

$$
\frac{\Psi, y : \tau; \Gamma; \Delta \Rightarrow A}{\Psi; \Gamma; \Delta \Rightarrow \forall y : \tau. A} \forall R
$$

- Left rule: matching use of service

$$
\frac{\forall \psi \vdash M : \tau \quad \Psi; \Gamma; \Delta', A\{M/y\} \Rightarrow C}{\Psi; \Gamma; \Delta', \forall y : \tau. A \Rightarrow C} \forall L
$$

- Proof reduction
Offering term input ($\forall y: \tau. A$)

- $P :: x : \forall y: \tau. A$
  - $P$ inputs an $M : \tau$ along $x$ and then behaves as $A\{M/x\}$

- **Right rule: offer of service**
  \[
  \begin{align*}
  &\psi, \, y: \tau ; \Gamma ; \Delta \Rightarrow x : A \\
  &\psi ; \Gamma ; \Delta \Rightarrow x : \forall y: \tau. A \quad \forall R
  \end{align*}
  \]

- **Left rule: matching use of service**
  \[
  \begin{align*}
  &\psi \vdash M : \tau \\
  &\psi ; \Gamma ; \Delta', \ A\{M/y\} \Rightarrow C \quad \forall L
  \end{align*}
  \]

- Proof reduction
Offering term input ($\forall y:\tau. A$)

- $P :: x : \forall y:\tau. A$
  - $P$ inputs an $M : \tau$ along $x$ and then behaves as $A\{M/x\}$
- Right rule: offer of service
  
  $\Psi, y:\tau ; \Gamma ; \Delta \Rightarrow P :: x : A$
  
  $\Psi ; \Gamma ; \Delta \Rightarrow x(y).P :: x : \forall y:\tau. A$

- Left rule: matching use of service

  $\Psi \vdash M : \tau \quad \Psi ; \Gamma ; \Delta', A\{M/y\} \Rightarrow C$
  
  $\Psi ; \Gamma ; \Delta', \forall y:\tau. A \Rightarrow C$

- Proof reduction
Offering term input ($\forall y: \tau. A$)

- $P :: x : \forall y : \tau. A$
  - $P$ inputs an $M : \tau$ along $x$ and then behaves as $A\{M/x\}$
- Right rule: offer of service
  \[
  \frac{\psi, y: \tau ; \Gamma ; \Delta \Rightarrow P :: x : A}{\psi ; \Gamma ; \Delta \Rightarrow x(y). P :: x : \forall y : \tau. A} \quad \forall R
  \]
- Left rule: matching use of service
  \[
  \frac{\psi \vdash M : \tau \quad \psi ; \Gamma ; \Delta', x: A\{M/y\} \Rightarrow z : C}{\psi ; \Gamma ; \Delta', x: \forall y : \tau. A \Rightarrow z : C} \quad \forall L
  \]
- Proof reduction
Offering term input ($\forall y : \tau. A$)

- $P :: x : \forall y : \tau. A$
  - $P$ inputs an $M : \tau$ along $x$ and then behaves as $A\{M/x\}$
- **Right rule: offer of service**
  
  \[
  \frac{
  \Psi, y : \tau ; \Gamma ; \Delta \Rightarrow P :: x : A
  }{
  \Psi ; \Gamma ; \Delta \Rightarrow x(y).P :: x : \forall y : \tau. A}
  \]

- **Left rule: matching use of service**
  
  \[
  \frac{
  \Psi \vdash M : \tau \quad \Psi ; \Gamma ; \Delta', x : A\{M/y\} \Rightarrow Q :: z : C
  }{
  \Psi ; \Gamma ; \Delta', x : \forall y : \tau. A \Rightarrow \bar{x}\langle M\rangle.Q :: z : C}
  \]

- **Proof reduction**
Offering term input ($\forall y : \tau . A$)

- $P :: x : \forall y : \tau . A$
  - $P$ inputs an $M : \tau$ along $x$ and then behaves as $A\{M/x\}$
- Right rule: offer of service
  \[
  \frac{\Psi, y : \tau ; \Gamma ; \Delta \Rightarrow P :: x : A}{\Psi ; \Gamma ; \Delta \Rightarrow x(y).P :: x : \forall y : \tau . A} \quad \forall R
  \]
- Left rule: matching use of service
  \[
  \frac{\Psi \vdash M : \tau \quad \Psi ; \Gamma ; \Delta', x : A\{M/y\} \Rightarrow Q :: z : C}{\Psi ; \Gamma ; \Delta', x : \forall y : \tau . A \Rightarrow \bar{x}\langle M\rangle . Q :: z : C} \quad \forall L
  \]
- Proof reduction
  \[
  (\nu x)(x(y).P | \bar{x}\langle M\rangle . Q) \rightarrow (\nu x)(P\{M/y\} | Q)
  \]
Term passing: other connectives

- Quantified proposition as dependent session types

  \[ x : \forall y : \tau. A \quad \text{Input an } M : A \text{ along } x \text{ and behave as } A\{M/y\} \]
  \[ x : \tau \rightarrow A \quad \text{Input an } M : A \text{ along } x \text{ and behave as } A \]
  \[ x : \exists y : \tau. A \quad \text{Output an } M : A \text{ along } x \text{ and behave as } A\{M/y\} \]
  \[ x : \tau \otimes A \quad \text{Output an } M : A \text{ along } x \text{ and behave as } A \]

- \( \tau \rightarrow A \) as shorthand for \( \forall y : \tau. A \) if \( y \) not free in \( A \)
- \( \tau \otimes A \) as shorthand for \( \exists y : \tau. A \) if \( y \) not free in \( A \)
- We will omit the ‘\$’ for readability
Examples, carrying proofs

- PDF indexing service

\[
\text{index}_3 : !(\text{file} \rightarrow \text{file} \otimes 1)
\]
\[
\text{index}_4 : !(\forall f:\text{file}. \text{pdf}(f) \rightarrow \exists g:\text{file}. \text{pdf}(g) \otimes 1)
\]

Persistently offer to input a file \(f\), a proof that \(f\) is in PDF format, then output a PDF file \(g\), and a proof that \(g\) is in PDF format and terminate the session.

- Persistent file storage

\[
\text{store}_1 : !(\text{file} \rightarrow !(\text{file} \otimes 1))
\]
\[
\text{store}_2 : !(\forall f:\text{file}. \exists g:\text{file}. g \equiv f \otimes 1)
\]

Persistently offer to input a file, then output a persistent channel for retrieving this file and a proof that the two are equal.
Outline

1. Session types for $\pi$-calculus
2. Dependent session types
3. Proof irrelevance
4. Some results
5. Conclusion
Proof irrelevance

- In many examples, we want to know that proofs exist, but we do not want to transmit them
  - We can easily check pdf\((g)\) when using the indexing service
  - The proof of \(g \vdash f\) (by reflexivity) would not be informative
- Use proof irrelevance in type theory
- \(M : [\tau] \rightarrow M\) is a term of type \(\tau\) that is computationally irrelevant
Proof irrelevance: rules

- Introduction and elimination
  \[
  \frac{\Psi \vdash M : \tau}{\Psi \vdash [M] : [\tau]} \quad [I] \\
  \frac{\Psi \vdash M : [\tau]}{\Psi, x : \tau \vdash N : \sigma} \quad [E]
  \]

- \(\Psi^\oplus\) promotes hypotheses \(x : \tau\) to \(x : \tau\)

- In examples, may use pattern matching instead of \texttt{let}

- By agreement, terms \([M]\) will be erased before transmission

- Typing guarantees this can be done consistently
Examples with proof irrelevance

- Mark proofs as computationally irrelevant
- PDF indexing service

\[ \text{index}_4 : \neg (\forall f : \text{file}. \text{pdf}(f) \rightarrow \exists g : \text{file}. \text{pdf}(g) \otimes 1) \]
\[ \text{index}_5 : \neg (\forall f : \text{file}. [\text{pdf}(f)] \rightarrow \exists g : \text{file}. [\text{pdf}(g)] \otimes 1) \]

- Persistent file storage

\[ \text{store}_2 : \neg (\forall f : \text{file}. \forall g : \text{file}. g \equiv f \otimes 1) \]
\[ \text{store}_3 : \neg (\forall f : \text{file}. \exists g : \text{file}. [g \equiv f] \otimes 1) \]

- After erasure, communication can be optimized further
Example: mobile code

- For sensitive documents we want to run indexing locally
- Specification

\[
\text{index}_5 : !((\Pi f:\text{file. [pdf}(f)\text{]} \rightarrow \Sigma g:\text{file. [pdf}(g)\text{]})) \otimes 1)
\]

- Service persistently offers a function for indexing
- Cannot leak information since only process layer can communicate
Outline

1. Session types for $\pi$-calculus
2. Dependent session types
3. Proof irrelevance
4. Some results
5. Conclusion
Some results

- Recall: typing is modulo (shallow) structural congruence
- Theorem: type preservation = session fidelity
- Theorem: progress = deadlock freedom
- Theorem: termination
  - Via linear logical relations (Pérez et al., ESOP 2012)
  - Some commuting conversions = behavioral equivalences
- Theorem: proof reduction = process reduction
  - Permuting cut and cut! = structural equivalences
  - Identity elimination = structural reduction (forwarding)
  - Propositional reduction = communication
  - Some commuting conversion needed if promotion is suppressed and termination is asynchronous
Further extensions and results

- Family of monads $\diamond_{K^T} = \text{digital signatures}$
  - Continuum of trust: from proofs to digital signatures (CPP 2011)
- Functions as session-typed processes (FoSSaCS 2012)
  - Translate from natural deduction to sequent calculus
  - Via linear natural deduction
  - $[T \rightarrow S] = ![T] \overset{\circ}{\rightarrow} [S]$: copying evaluation (by name)
  - $(T \rightarrow S)^* = ![T^* \overset{\circ}{\rightarrow} S^*]$: sharing evaluation (futures)
  - By-value and by-need are particular schedules for sharing
Ongoing work

- Polymorphism
  - Immediate in the functional layer
  - Parametricity in the process layer (Pérez et al., ESOP 2012)

- Asynchronous session types (w. Henry DeYoung)
  - Also via Curry-Howard isomorphism!
  - Unlocking parallelism with commuting conversions
  - Each channel implementable as a bidirectional queue

- Classical linear logic
  - Superficially more economical
  - Does not enforce locality of shared channels
  - All standard session examples (and more) already expressible in intuitionistic system
  - Less likely to lead to full type theory
Concurrent dependent type theory?

- At present, we have a two-layer system
  - Communication layer (both linear and shared channels)
  - Value layer (dependent type theory)
- Can we have a concurrent dependently-typed language?
  - Problem of linear dependency
  - Equational reasoning about processes
  - Integrating natural deduction and sequent calculus
  - Dependently typed functional translation?
  - Monadic encapsulation, à la CLF?
Some related work

- Computational interpretations of linear logic (Abramsky 1993)
- Relating $\pi$-calculus and linear logic (Bellin & Scott 1994)
- Session types (Honda 1993) (Honda et al. 1998) . . .
- Lolliproc (Mazurak & Zdancewic 2010)
  - Natural deduction for classical linear logic
  - Purely linear (unrestricted version conjectured)
  - Tighter integration of functions with processes
  - Requires control operators and additional coercions
  - Dependent version unlikely?
A Curry-Howard isomorphism

- Linear propositions as session types
  \( A \to B \) (input), \( A \otimes B \) (output), \( 1 \) (termination)
  \( A \& B \) (external choice), \( A \oplus B \) (internal choice), \(!A\) (replication)

- Sequent proofs as \( \pi \)-calculus processes with a binary guarded choice and channel forwarding

- Cut reduction as \( \pi \)-calculus reduction

Term-passing extension with a type theory

- \( \forall x:\tau. A \) (term input), \( \exists x:\tau. A \) (term output)

Additional type theory constructs

- \([\tau]\) for proof irrelevance (not transmitted)
- \(\diamondsuit_{K}\tau\) for affirmations (evidenced by digital signatures)