Towards Concurrent Type Theory

Luís Caires[1], Frank Pfenning[2], Bernardo Toninho[1,2]

[1] Universidade Nova de Lisboa
[2] Carnegie Mellon University

Workshop on Types in Language Design
and Implementation (TLDI)
January 28, 2012

Proofs and programs

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

Proofs and programs

- 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 λ-terms
  - Proof reduction is functional computation

Proofs and programs

- 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 λ-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 π-calculus processes
  - Cut reduction as communication

Type theory

- 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

Judgment forms

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

$$\underbrace{A_1, \ldots, A_n}_{\Delta} \Rightarrow A$$

- Cut as composition

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

- Identity as forwarding

$$\frac{}{A \Rightarrow A} \ \mathsf{id}_A$$

Judgment forms

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

$$\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Delta} \Rightarrow P :: x : A$$

  P uses $x_i{:}A_i$ and offers $x{:}A$.
- Cut as composition

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

- Identity as forwarding

$$\frac{}{A \Rightarrow A}\ \mathsf{id}_A$$

Judgment forms

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

$$\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Delta} \Rightarrow P :: x : A$$

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

- Cut as composition

$$\frac{\Delta \Rightarrow \quad x : A \qquad \Delta', x{:}A \Rightarrow \quad z : C}{\Delta, \Delta' \Rightarrow \quad z : C}\ \mathrm{cut}_A$$

- Identity as forwarding

$$\frac{}{A \Rightarrow A}\ \mathrm{id}_A$$

Judgment forms

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

$$\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Delta} \Rightarrow P :: x : A$$

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

- Cut as composition

$$\frac{\Delta \Rightarrow P :: x : A \qquad \Delta', x{:}A \Rightarrow Q :: z : C}{\Delta, \Delta' \Rightarrow (\nu x)(P \mid Q) :: z : C}\ \mathsf{cut}_A$$

- Identity as forwarding

$$\frac{}{A \Rightarrow A}\ \mathsf{id}_A$$

Judgment forms

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

$$\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Delta} \Rightarrow P :: x : A$$

  P uses $x_i{:}A_i$ and offers $x{:}A$.
- Cut as composition

$$\frac{\Delta \Rightarrow P :: x : A \qquad \Delta', x{:}A \Rightarrow Q :: z : C}{\Delta, \Delta' \Rightarrow (\nu x)(P \mid Q) :: z : C}\ \mathrm{cut}_A$$

- Identity as forwarding

$$\frac{}{x{:}A \Rightarrow \qquad\qquad z : A}\ \mathrm{id}_A$$

Judgment forms

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

$$\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Delta} \Rightarrow P :: x : A$$

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

- Cut as composition

$$\frac{\Delta \Rightarrow P :: x : A \qquad \Delta', x{:}A \Rightarrow Q :: z : C}{\Delta, \Delta' \Rightarrow (\nu x)(P \mid Q) :: z : C}\ \mathrm{cut}_A$$

- Identity as forwarding

$$\frac{}{x{:}A \Rightarrow [x \leftrightarrow z] :: z : A}\ \mathrm{id}_A$$

Offering input ($A \multimap B$)

- $P :: x : A \multimap B$
  - P inputs an A along x and then behaves as B
- Right rule: offer of service

$$\frac{\Delta, A \Rightarrow B}{\Delta \Rightarrow A \multimap B} \multimap R$$

- Left rule: matching use of service

$$\frac{\Delta \Rightarrow A \quad \Delta', B \Rightarrow C}{\Delta, \Delta', A \multimap B \Rightarrow C} \multimap L$$

Offering input ($A \multimap B$)

- $P :: x : A \multimap B$
  - P inputs an A along x and then behaves as B
- Right rule: offer of service

$$\frac{\Delta, \quad A \Rightarrow \qquad\qquad B}{\Delta \Rightarrow \qquad\qquad\qquad A \multimap B} \multimap R$$

- Left rule: matching use of service

$$\frac{\Delta \Rightarrow A \quad \Delta', B \Rightarrow C}{\Delta, \Delta', A \multimap B \Rightarrow C} \multimap L$$

Offering input ($A \multimap B$)

- $P :: x : A \multimap B$
  - P inputs an A along x and then behaves as B
- Right rule: offer of service

$$\frac{\Delta, y{:}A \Rightarrow \quad x : B}{\Delta \Rightarrow \quad x : A \multimap B} \multimap R$$

- Left rule: matching use of service

$$\frac{\Delta \Rightarrow A \quad \Delta', B \Rightarrow C}{\Delta, \Delta', A \multimap B \Rightarrow C} \multimap L$$

Offering input ($A \multimap B$)

- $P :: x : A \multimap B$
  - P inputs an A along x and then behaves as B
- Right rule: offer of service

$$\frac{\Delta, y{:}A \Rightarrow P :: x : B}{\Delta \Rightarrow x(y).P :: x : A \multimap B} \multimap R$$

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

$$\frac{\Delta \Rightarrow A \quad \Delta', B \Rightarrow C}{\Delta, \Delta', A \multimap B \Rightarrow C} \multimap L$$

Offering input ($A \multimap B$)

- $P :: x : A \multimap B$
  - P inputs an A along x and then behaves as B
- Right rule: offer of service

$$\frac{\Delta, y{:}A \Rightarrow P :: x : B}{\Delta \Rightarrow x(y).P :: x : A \multimap B} \multimap R$$

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

$$\frac{\Delta \Rightarrow \qquad A \qquad \Delta', \quad B \Rightarrow \qquad C}{\Delta, \Delta', \quad A \multimap B \Rightarrow \qquad C} \multimap L$$

Offering input ($A \multimap B$)

- $P :: x : A \multimap B$
  - P inputs an A along x and then behaves as B
- Right rule: offer of service

$$\frac{\Delta, y{:}A \Rightarrow P :: x : B}{\Delta \Rightarrow x(y).P :: x : A \multimap B} \multimap R$$

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

$$\frac{\Delta \Rightarrow \quad y : A \qquad \Delta', x{:}B \Rightarrow \quad z : C}{\Delta, \Delta', x{:}A \multimap B \Rightarrow \quad z : C} \multimap L$$

Offering input ($A \multimap B$)

- $P :: x : A \multimap B$
  - P inputs an A along x and then behaves as B
- Right rule: offer of service

$$\frac{\Delta, y{:}A \Rightarrow P :: x : B}{\Delta \Rightarrow x(y).P :: x : A \multimap B} \multimap R$$

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

$$\frac{\Delta \Rightarrow P :: y : A \qquad \Delta', x{:}B \Rightarrow Q :: z : C}{\Delta, \Delta', x{:}A \multimap B \Rightarrow (\boldsymbol{\nu} y)\overline{x}\langle y\rangle.(P \mid Q) :: z : C} \multimap L$$

  - Can reuse x, due to linearity
  - Channel y must be new (bound output)

Proof and process reduction

- Proof reduction

$$\dfrac{\dfrac{\Delta, A \Rightarrow B}{\Delta \Rightarrow A \multimap B}\ {\multimap}R \qquad \dfrac{\Delta_1 \Rightarrow A \quad \Delta_2, B \Rightarrow C}{\Delta_1, \Delta_2, A \multimap B \Rightarrow C}\ {\multimap}L}{\Delta, \Delta_1, \Delta_2 \Rightarrow C}\ \mathrm{cut}_{A \multimap B}$$

$$\longrightarrow$$

$$\dfrac{\dfrac{\Delta_1 \Rightarrow A \quad \Delta, A \Rightarrow B}{\Delta, \Delta_1 \Rightarrow B}\ \mathrm{cut}_A \qquad \Delta_2, B \Rightarrow C}{\Delta, \Delta_1, \Delta_2 \Rightarrow C}\ \mathrm{cut}_B$$

- Corresponding process reduction

$$\Delta, \Delta_1, \Delta_2 \Rightarrow (\boldsymbol{\nu} x)(x(y).P_1 \mid (\boldsymbol{\nu} w)(\overline{x}\langle w\rangle.(P_2 \mid Q))) :: z : C$$

$$\longrightarrow$$

$$\Delta, \Delta_1, \Delta_2 \Rightarrow (\boldsymbol{\nu} x)((\boldsymbol{\nu} w)(P_2 \mid P_1\{w/y\}) \mid Q) :: z : C$$

Process reduction

- Corresponding process reduction

$$(\boldsymbol{\nu} x)(x(y).P_1 \mid (\boldsymbol{\nu} w)(\overline{x}\langle w\rangle.(P_2 \mid Q)))$$
$$\longrightarrow$$
$$(\boldsymbol{\nu} x)((\boldsymbol{\nu} w)(P_2 \mid P_1\{w/y\}) \mid Q)$$

- Instance of (modulo structural congruence)

$$(x(y).P \mid \overline{x}\langle w\rangle.Q) \longrightarrow (P\{w/y\} \mid Q)$$

- Synchronous π-calculus
- Typing modulo structural congruence

Preview

- Linear propositions as session types

$P :: x : A \multimap B$	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
$P :: x : \mathbf{1}$	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

$$\frac{\Delta \Rightarrow \quad A \qquad \Delta' \Rightarrow \quad B}{\Delta, \Delta' \Rightarrow \quad A \otimes B} \otimes R$$

- 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

$$\frac{\Delta \Rightarrow \quad y : A \qquad \Delta' \Rightarrow \quad x : B}{\Delta, \Delta' \Rightarrow \quad x : A \otimes B} \otimes R$$

- 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

$$\frac{\Delta \Rightarrow P :: y : A \qquad \Delta' \Rightarrow Q :: x : B}{\Delta, \Delta' \Rightarrow (\boldsymbol{\nu} y)\overline{x}\langle y \rangle.(P \mid Q) :: x : A \otimes B} \otimes R$$

- 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

$$\frac{\Delta \Rightarrow P :: y : A \qquad \Delta' \Rightarrow Q :: x : B}{\Delta, \Delta' \Rightarrow (\boldsymbol{\nu} y)\overline{x}\langle y \rangle.(P \mid Q) :: x : A \otimes B} \otimes R$$

- Left rule: perform matching input

$$\frac{\Delta, \quad A, \quad B \Rightarrow \qquad\qquad C}{\Delta, \quad A \otimes B \Rightarrow \qquad\qquad C} \otimes L$$

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

$$\dfrac{\Delta \Rightarrow P :: y : A \qquad \Delta' \Rightarrow Q :: x : B}{\Delta, \Delta' \Rightarrow (\boldsymbol{\nu} y)\overline{x}\langle y\rangle.(P \mid Q) :: x : A \otimes B}\ \otimes R$$

- Left rule: perform matching input

$$\dfrac{\Delta, y{:}A, x{:}B \Rightarrow \qquad z : C}{\Delta, x{:}A \otimes B \Rightarrow \qquad z : C}\ \otimes L$$

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

$$\frac{\Delta \Rightarrow P :: y : A \qquad \Delta' \Rightarrow Q :: x : B}{\Delta, \Delta' \Rightarrow (\boldsymbol{\nu} y)\overline{x}\langle y \rangle.(P \mid Q) :: x : A \otimes B} \otimes R$$

- Left rule: perform matching input

$$\frac{\Delta, y{:}A, x{:}B \Rightarrow P :: z : C}{\Delta, x{:}A \otimes B \Rightarrow x(y).P :: z : C} \otimes L$$

Offering output ($A \otimes B$)

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

  $x{:}A{\otimes}B \Rightarrow x(y).(\boldsymbol{\nu} w)\overline{z}\langle w\rangle.([x \leftrightarrow w] \mid [y \leftrightarrow z]) :: z : B{\otimes}A$

Termination (1)

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

$$\frac{}{\cdot \Rightarrow \mathbf{1}}\ \mathbf{1}R$$

- Left rule: accept termination

$$\frac{\Delta \Rightarrow C}{\Delta, \mathbf{1} \Rightarrow C}\ \mathbf{1}L$$

- Reduction

Termination (1)

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

$$\frac{}{\cdot \Rightarrow \mathbf{1}}\ \mathbf{1}R$$

- Left rule: accept termination

$$\frac{\Delta \Rightarrow C}{\Delta, \mathbf{1} \Rightarrow C}\ \mathbf{1}L$$

- Reduction

Termination (1)

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

$$\frac{}{\cdot \Rightarrow \qquad\qquad x : \mathbf{1}}\ \mathbf{1}R$$

- Left rule: accept termination

$$\frac{\Delta \Rightarrow C}{\Delta, \mathbf{1} \Rightarrow C}\ \mathbf{1}L$$

- Reduction

Termination (1)

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

$$\frac{}{\cdot \Rightarrow \overline{x}\langle\rangle.\mathbf{0} :: x : \mathbf{1}}\ \mathbf{1}R$$

- Left rule: accept termination

$$\frac{\Delta \Rightarrow C}{\Delta, \mathbf{1} \Rightarrow C}\ \mathbf{1}L$$

- Reduction

Termination (1)

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

$$\frac{}{\cdot \Rightarrow \overline{x}\langle\rangle.\mathbf{0} :: x : \mathbf{1}}\ \mathbf{1}R$$

- Left rule: accept termination

$$\frac{\Delta \Rightarrow \qquad\qquad C}{\Delta,\quad \mathbf{1} \Rightarrow \qquad\qquad C}\ \mathbf{1}L$$

- Reduction

Termination (1)

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

$$\frac{}{\cdot \Rightarrow \overline{x}\langle\rangle.\mathbf{0} :: x : \mathbf{1}}\ \mathbf{1}R$$

- Left rule: accept termination

$$\frac{\Delta \Rightarrow \qquad z : C}{\Delta, x{:}\mathbf{1} \Rightarrow \qquad z : C}\ \mathbf{1}L$$

- Reduction

Termination (1)

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

$$\frac{}{\cdot \Rightarrow \overline{x}\langle\rangle.\mathbf{0} :: x : \mathbf{1}}\ \mathbf{1}R$$

- Left rule: accept termination

$$\frac{\Delta \Rightarrow P :: z : C}{\Delta, x{:}\mathbf{1} \Rightarrow x().P :: z : C}\ \mathbf{1}L$$

- Reduction

Termination (1)

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

$$\frac{}{\cdot \Rightarrow \color{red}{\overline{x}\langle\rangle.\mathbf{0}} :: x : \mathbf{1}}\ \mathbf{1}R$$

- Left rule: accept termination

$$\frac{\Delta \Rightarrow \color{red}{P} :: z : C}{\Delta, x{:}\mathbf{1} \Rightarrow \color{red}{x().P} :: z : C}\ \mathbf{1}L$$

- Reduction

$$(\overline{x}\langle\rangle.\mathbf{0} \mid x().P) \longrightarrow P$$

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

$$\mathsf{index}_1 \triangleq \mathsf{file} \multimap (\mathsf{file} \otimes \mathbf{1})$$

- Shape of a server

$$\mathsf{srv} \triangleq x(f).(\boldsymbol{\nu} y)\overline{x}\langle y\rangle.(P \mid \overline{x}\langle\rangle.\mathbf{0}) :: x : \mathsf{index}_1$$

- Shape of a client

$$\mathsf{client} \triangleq (\boldsymbol{\nu} pdf)\overline{x}\langle pdf\rangle.x(idx).x().Q$$

- Composition of server and client

$$\frac{\cdot \Rightarrow \mathsf{srv} :: x : \mathsf{index}_1 \quad x : \mathsf{index}_1 \Rightarrow \mathsf{client} :: z : \mathbf{1}}{\cdot \Rightarrow (\nu x)(\mathsf{srv} \mid \mathsf{client}) :: z : \mathbf{1}}\ \mathsf{cut}$$

Taking stock I

- At this point we have

$$
\begin{array}{llcll}
\textit{Types} & A, B, C & ::= & A \multimap B & \text{input} \\
 & & \mid & A \otimes B & \text{output} \\
 & & \mid & \mathbf{1} & \text{termination} \\
\textit{Processes} & P, Q & ::= & [x \leftrightarrow z] & \text{forwarding} \\
 & & \mid & (P \mid Q) & \text{parallel composition} \\
 & & \mid & (\boldsymbol{\nu} x)P & \text{name restriction} \\
 & & \mid & x(y).P & \text{input} \\
 & & \mid & (\nu y)\overline{x}\langle y\rangle.P & \text{bound output} \\
 & & \mid & x().P & \text{(wait)} \\
 & & \mid & \overline{x}\langle\rangle.\mathbf{0} & \text{(termination)}
\end{array}
$$

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

$$\frac{\Delta \Rightarrow A \quad \Delta \Rightarrow B}{\Delta \Rightarrow A \,\&\, B} \;\&R$$

- Left rules: making a choice between A and B

$$\frac{\Delta, A \Rightarrow C}{\Delta, A \,\&\, B \Rightarrow C} \;\&L_1$$

$$\frac{\Delta, B \Rightarrow C}{\Delta, A \,\&\, B \Rightarrow C} \;\&L_2$$

Offering external choice ($A \mathbin{\&} B$)

- $P :: x : A \mathbin{\&} B$
  - P offers the choice between A and B along x
- Right rule: offering choice between A and B

$$\frac{\Delta \Rightarrow \qquad\qquad A \qquad \Delta \Rightarrow \qquad\qquad B}{\Delta \Rightarrow \qquad\qquad\qquad\qquad A \mathbin{\&} B} \mathbin{\&}R$$

- Left rules: making a choice between A and B

$$\frac{\Delta, A \Rightarrow C}{\Delta, A \mathbin{\&} B \Rightarrow C} \mathbin{\&}L_1$$

$$\frac{\Delta, B \Rightarrow C}{\Delta, A \mathbin{\&} B \Rightarrow C} \mathbin{\&}L_2$$

Offering external choice ($A \mathbin{\&} B$)

- $P :: x : A \mathbin{\&} B$
  - P offers the choice between A and B along x
- Right rule: offering choice between A and B

$$\frac{\Delta \Rightarrow \qquad x : A \qquad\qquad \Delta \Rightarrow \qquad x : B}{\Delta \Rightarrow \qquad\qquad\qquad x : A \mathbin{\&} B} \;\&R$$

- Left rules: making a choice between A and B

$$\frac{\Delta, A \Rightarrow C}{\Delta, A \mathbin{\&} B \Rightarrow C} \;\&L_1$$

$$\frac{\Delta, B \Rightarrow C}{\Delta, A \mathbin{\&} B \Rightarrow C} \;\&L_2$$

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

$$\frac{\Delta \Rightarrow P :: x : A \qquad \Delta \Rightarrow Q :: x : B}{\Delta \Rightarrow x.\mathsf{case}(P, Q) :: x : A \,\&\, B} \,\&R$$

- Left rules: making a choice between A and B

$$\frac{\Delta, A \Rightarrow C}{\Delta, A \,\&\, B \Rightarrow C} \,\&L_1$$

$$\frac{\Delta, B \Rightarrow C}{\Delta, A \,\&\, B \Rightarrow C} \,\&L_2$$

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

$$\frac{\Delta \Rightarrow P :: x : A \qquad \Delta \Rightarrow Q :: x : B}{\Delta \Rightarrow x.\mathrm{case}(P, Q) :: x : A \& B} \ \&R$$

- Left rules: making a choice between A and B

$$\frac{\Delta, \quad A \Rightarrow \qquad C}{\Delta, \quad A \& B \Rightarrow \qquad C} \ \&L_1$$

$$\frac{\Delta, \quad B \Rightarrow \qquad C}{\Delta, \quad A \& B \Rightarrow \qquad C} \ \&L_2$$

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

$$\frac{\Delta \Rightarrow P :: x : A \qquad \Delta \Rightarrow Q :: x : B}{\Delta \Rightarrow x.\mathsf{case}(P, Q) :: x : A \,\&\, B}\ \&R$$

- Left rules: making a choice between A and B

$$\frac{\Delta, x{:}A \Rightarrow \qquad z : C}{\Delta, x{:}A \,\&\, B \Rightarrow \qquad z : C}\ \&L_1$$

$$\frac{\Delta, x{:}B \Rightarrow \qquad z : C}{\Delta, x{:}A \,\&\, B \Rightarrow \qquad z : C}\ \&L_2$$

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

$$\frac{\Delta \Rightarrow P :: x : A \qquad \Delta \Rightarrow Q :: x : B}{\Delta \Rightarrow x.\mathrm{case}(P, Q) :: x : A \,\&\, B}\ \&R$$

- Left rules: making a choice between A and B

$$\frac{\Delta, x{:}A \Rightarrow Q :: z : C}{\Delta, x{:}A \,\&\, B \Rightarrow x.\mathrm{inl};\, Q :: z : C}\ \&L_1$$

$$\frac{\Delta, x{:}B \Rightarrow Q :: z : C}{\Delta, x{:}A \,\&\, B \Rightarrow x.\mathrm{inr};\, Q :: z : C}\ \&L_2$$

Offering external choice ($A \,\&\, B$)

- Need binary guarded choice construct
- New reductions

$$(x.\mathsf{case}(P, Q) \mid x.\mathsf{inl}; R) \longrightarrow (P \mid R)$$
$$(x.\mathsf{case}(P, Q) \mid x.\mathsf{inr}; R) \longrightarrow (Q \mid R)$$

Example: PDF compression

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

$$
\begin{aligned}
\mathsf{server}_1 \quad &\triangleq \quad (\mathsf{file} \multimap (\mathsf{file} \otimes \mathbf{1})) \\
&\qquad \& \; (\mathsf{file} \multimap (\mathsf{file} \otimes \mathbf{1}))
\end{aligned}
$$

- Different protocol: decision is made later

$$\mathsf{server}_2 \quad \triangleq \quad \mathsf{file} \multimap ((\mathsf{file} \,\&\, \mathsf{file}) \otimes \mathbf{1})$$

- In practice, should use labeled products $\&_i\{l_i : A_i\}$

Offering internal choice ($A \oplus B$)

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

$$\frac{\Delta \Rightarrow A}{\Delta \Rightarrow A \oplus B} \oplus R_1$$

$$\frac{\Delta \Rightarrow B}{\Delta \Rightarrow A \oplus B} \oplus R_2$$

- Accounting for either A or B:

$$\frac{\Delta, A \Rightarrow C \quad \Delta, B \Rightarrow C}{\Delta, A \oplus B \Rightarrow C} \oplus L$$

- 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:

$$\frac{\Delta \Rightarrow A}{\Delta \Rightarrow A \oplus B}\ \oplus R_1$$

$$\frac{\Delta \Rightarrow B}{\Delta \Rightarrow A \oplus B}\ \oplus R_2$$

- Accounting for either A or B:

$$\frac{\Delta, A \Rightarrow C \quad \Delta, B \Rightarrow C}{\Delta, A \oplus B \Rightarrow C}\ \oplus L$$

- 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:

$$\frac{\Delta \Rightarrow \quad x : A}{\Delta \Rightarrow \quad x : A \oplus B} \oplus R_1$$

$$\frac{\Delta \Rightarrow \quad x : B}{\Delta \Rightarrow \quad x : A \oplus B} \oplus R_2$$

- Accounting for either A or B:

$$\frac{\Delta, A \Rightarrow C \quad \Delta, B \Rightarrow C}{\Delta, A \oplus B \Rightarrow C} \oplus L$$

- 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:

$$\frac{\Delta \Rightarrow P :: x : A}{\Delta \Rightarrow x.\mathrm{inl}; P :: x : A \oplus B} \oplus R_1$$

$$\frac{\Delta \Rightarrow P :: x : B}{\Delta \Rightarrow x.\mathrm{inr}; P :: x : A \oplus B} \oplus R_2$$

- Accounting for either A or B:

$$\frac{\Delta, A \Rightarrow C \quad \Delta, B \Rightarrow C}{\Delta, A \oplus B \Rightarrow C} \oplus L$$

- 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:

$$\frac{\Delta \Rightarrow P :: x : A}{\Delta \Rightarrow x.\mathrm{inl}; P :: x : A \oplus B} \oplus R_1$$

$$\frac{\Delta \Rightarrow P :: x : B}{\Delta \Rightarrow x.\mathrm{inr}; P :: x : A \oplus B} \oplus R_2$$

- Accounting for either A or B:

$$\frac{\Delta, \quad A \Rightarrow \qquad C \qquad \Delta, \quad B \Rightarrow \qquad C}{\Delta, \quad A \oplus B \Rightarrow \qquad C} \oplus L$$

- 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:

$$\frac{\Delta \Rightarrow P :: x : A}{\Delta \Rightarrow x.\mathrm{inl}; P :: x : A \oplus B} \oplus R_1$$

$$\frac{\Delta \Rightarrow P :: x : B}{\Delta \Rightarrow x.\mathrm{inr}; P :: x : A \oplus B} \oplus R_2$$

- Accounting for either A or B:

$$\frac{\Delta, x{:}A \Rightarrow \quad z : C \qquad \Delta, x{:}B \Rightarrow \quad z : C}{\Delta, x{:}A \oplus B \Rightarrow \qquad z : C} \oplus L$$

- 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:

$$\frac{\Delta \Rightarrow P :: x : A}{\Delta \Rightarrow x.\mathrm{inl}; P :: x : A \oplus B} \oplus R_1$$

$$\frac{\Delta \Rightarrow P :: x : B}{\Delta \Rightarrow x.\mathrm{inr}; P :: x : A \oplus B} \oplus R_2$$

- Accounting for either A or B:

$$\frac{\Delta, x{:}A \Rightarrow P :: z : C \qquad \Delta, x{:}B \Rightarrow Q :: z : C}{\Delta, x{:}A \oplus B \Rightarrow x.\mathrm{case}(P, Q) :: z : C} \oplus L$$

- No new reductions

Example: PDF indexing

- Offer to index PDF, or indicate failure

$$\mathsf{index}_2 \quad \triangleq \quad (\mathsf{file} \multimap ((\mathsf{file} \otimes \mathbf{1}) \oplus \mathbf{1}))$$

- In practice, should use labeled sums $\oplus_i\{l_i : A_i\}$

Persistence

- To have persistent services, we generalize the judgment form (Hodas & Miller 1991; Andreoli 1992; Barber 1996)

$$\underbrace{B_1, \ldots, B_k}_{\substack{\Gamma \\ \text{persistently true}}} \quad ; \quad \underbrace{A_1, \ldots, A_n}_{\substack{\Delta \\ \text{linearly true}}} \Rightarrow C$$

- Label with shared channels u and linear channels x

$$\underbrace{u_1{:}B_1, \ldots, u_k{:}B_k}_{\substack{\Gamma \\ \text{shared}}} \; ; \; \underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\substack{\Delta \\ \text{linear}}} \Rightarrow P :: z : C$$

Structural rules

- cut! as composition with replicated input

$$\frac{\Gamma \,;\, \cdot \Rightarrow A \quad \Gamma, A \,;\, \Delta \Rightarrow C}{\Gamma \,;\, \Delta \Rightarrow C}\ \mathsf{cut!}_A$$

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

$$\frac{\Gamma, A \,;\, \Delta, A \Rightarrow C}{\Gamma, A \,;\, \Delta \Rightarrow C}\ \mathsf{copy}$$

Structural rules

- cut! as composition with replicated input

$$\frac{\Gamma\,;\,\cdot \Rightarrow A \qquad \Gamma, A\,;\,\Delta \Rightarrow C}{\Gamma\,;\,\Delta \Rightarrow C}\ \text{cut!}_A$$

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

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

Structural rules

- cut! as composition with replicated input

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

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

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

Structural rules

- cut! as composition with replicated input

$$\frac{\Gamma\,;\,\cdot \Rightarrow P :: x : A \qquad \Gamma, u{:}A\,;\,\Delta \Rightarrow Q :: z : C}{\Gamma\,;\,\Delta \Rightarrow (\boldsymbol{\nu} u)(!u(x).P \mid Q) :: z : C}\ \text{cut!}_A$$

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

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

Structural rules

- cut! as composition with replicated input

$$\frac{\Gamma \,;\, \cdot \Rightarrow P :: x : A \qquad \Gamma, u{:}A \,;\, \Delta \Rightarrow Q :: z : C}{\Gamma \,;\, \Delta \Rightarrow (\nu u)(!u(x).P \mid Q) :: z : C}\ \text{cut!}_A$$

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

$$\frac{\Gamma, \quad A \,;\, \Delta, \quad A \Rightarrow \qquad\qquad C}{\Gamma, \quad A \,;\, \Delta \Rightarrow \qquad\qquad C}\ \text{copy}$$

Structural rules

- cut! as composition with replicated input

$$\frac{\Gamma\,;\,\cdot \Rightarrow P :: x : A \qquad \Gamma, u{:}A\,;\,\Delta \Rightarrow Q :: z : C}{\Gamma\,;\,\Delta \Rightarrow (\boldsymbol{\nu} u)(!u(x).P \mid Q) :: z : C}\ \mathsf{cut!}_A$$

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

$$\frac{\Gamma, u{:}A\,;\,\Delta, y{:}A \Rightarrow \qquad z : C}{\Gamma, u{:}A\,;\,\Delta \Rightarrow \qquad z : C}\ \mathsf{copy}$$

Structural rules

- cut! as composition with replicated input

$$\frac{\Gamma\,;\,\cdot \Rightarrow P :: x : A \qquad \Gamma, u{:}A\,;\,\Delta \Rightarrow Q :: z : C}{\Gamma\,;\,\Delta \Rightarrow (\boldsymbol{\nu} u)(!u(x).P \mid Q) :: z : C}\ \mathsf{cut!}_A$$

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

$$\frac{\Gamma, u{:}A\,;\,\Delta, y{:}A \Rightarrow P :: z : C}{\Gamma, u{:}A\,;\,\Delta \Rightarrow (\boldsymbol{\nu} y)\overline{u}\langle y\rangle.P :: z : C}\ \mathsf{copy}$$

- y will be linear and behave according to A

Cut! reduction

- Replaying the proof reduction yields:

$$\begin{array}{rl} & (\boldsymbol{\nu} u)(!u(x).P \mid (\boldsymbol{\nu} y)\overline{u}\langle y\rangle.Q) \\ \longrightarrow & (\boldsymbol{\nu} y)(P\{y/x\} \mid (\boldsymbol{\nu} u)(!u(x).P \mid Q)) \end{array}$$

- 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

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

- Sharing a persistent service

$$\frac{\Gamma, A \,;\, \Delta \Rightarrow C}{\Gamma \,;\, \Delta, !A \Rightarrow C}\ !L$$

Offering persistent service ($!A$)

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

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

- Sharing a persistent service

$$\frac{\Gamma, A \,;\, \Delta \Rightarrow C}{\Gamma \,;\, \Delta, !A \Rightarrow C} \; !L$$

Offering persistent service ($!A$)

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

$$\frac{\Gamma \,;\, \cdot \Rightarrow \qquad y : A}{\Gamma \,;\, \cdot \Rightarrow \qquad\qquad x : !A}\ !R$$

- Sharing a persistent service

$$\frac{\Gamma, A \,;\, \Delta \Rightarrow C}{\Gamma \,;\, \Delta, !A \Rightarrow C}\ !L$$

Offering persistent service ($!A$)

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

$$\frac{\Gamma \,;\, \cdot \Rightarrow P :: y : A}{\Gamma \,;\, \cdot \Rightarrow !x(y).P :: x : !A}\ !R$$

- Sharing a persistent service

$$\frac{\Gamma, A \,;\, \Delta \Rightarrow C}{\Gamma \,;\, \Delta, !A \Rightarrow C}\ !L$$

Offering persistent service ($!A$)

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

$$\frac{\Gamma ; \cdot \Rightarrow P :: y : A}{\Gamma ; \cdot \Rightarrow !x(y).P :: x : !A} \; !R$$

- Sharing a persistent service

$$\frac{\Gamma, \quad A ; \Delta \Rightarrow \qquad\qquad C}{\Gamma ; \Delta, \quad !A \Rightarrow \qquad\qquad C} \; !L$$

Offering persistent service ($!A$)

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

$$\frac{\Gamma \, ; \cdot \Rightarrow P :: y : A}{\Gamma \, ; \cdot \Rightarrow\ !x(y).P :: x : !A} \ !R$$

- Sharing a persistent service

$$\frac{\Gamma, u{:}A \, ; \Delta \Rightarrow \qquad z : C}{\Gamma \, ; \Delta, x{:}!A \Rightarrow \qquad\qquad z : C} \ !L$$

Offering persistent service ($!A$)

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

$$\frac{\Gamma \,;\, \cdot \Rightarrow P :: y : A}{\Gamma \,;\, \cdot \Rightarrow !x(y).P :: x : !A}\ !R$$

- Sharing a persistent service

$$\frac{\Gamma, u{:}A \,;\, \Delta \Rightarrow Q :: z : C}{\Gamma \,;\, \Delta, x{:}!A \Rightarrow x/u.Q :: z : C}\ !L$$

Promotion

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

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

Example: persistent storage

- Persistent PDF indexing service

$$\mathsf{index}_3 : !(\mathsf{file} \multimap \mathsf{file} \otimes \mathbf{1})$$

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

- Store a file persistently

$$\mathsf{store}_1 : !(\mathsf{file} \multimap !(\mathsf{file} \otimes \mathbf{1}))$$

  Persistently offer to input a file, then output a persistent handle for retrieving this file.

Taking stock II

- At this point we have in addition

$$
\begin{array}{llcll}
\textit{Types} & A, B, C & ::= & \ldots & \\
 & & \mid & A \,\&\, B & \text{external choice} \\
 & & \mid & A \oplus B & \text{internal choice} \\
 & & \mid & !A & \text{replication} \\
\textit{Processes} & P, Q & ::= & \ldots & \\
 & & \mid & x.\mathrm{inl}; P \mid x.\mathrm{inr}; P & \text{selection} \\
 & & \mid & x.\mathrm{case}(P, Q) & \text{branching} \\
 & & \mid & !u(x).P & \text{replicating input} \\
 & & \mid & x/u.P & \text{(promotion)}
\end{array}
$$

Outline

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

Passing terms

- Types τ from a (dependent) type theory
- Hypothetical judgment $\underbrace{x_1{:}\tau_1, \ldots, x_k{:}\tau_k}_{\Psi} \vdash M : \tau$
- Some example type constructors

$\Pi x{:}\tau.\sigma$, $\tau \to \sigma$	Functions from τ to σ
$\Sigma x{:}\tau.\sigma$, $\tau \times \sigma$	Pairs of a τ and a σ
nat	Natural numbers

- Integrate into sequent calculus

$$\underbrace{\Psi}_{\text{term variables}} \; ; \; \underbrace{\Gamma}_{\text{shared channels}} \; ; \; \underbrace{\Delta}_{\text{linear channels}} \Rightarrow P :: \underbrace{x : A}_{\text{linear}}$$

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

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

- Left rule: matching use of service

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

- 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

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

- Left rule: matching use of service

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

- 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

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

- Left rule: matching use of service

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

- 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

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

- Left rule: matching use of service

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

- 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

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

- Left rule: matching use of service

$$\frac{\Psi \vdash M : \tau \quad \Psi \,;\, \Gamma \,;\, \Delta', x{:}A\{M/y\} \Rightarrow Q :: z : C}{\Psi \,;\, \Gamma \,;\, \Delta', x{:}\forall y{:}\tau.A \Rightarrow \overline{x}\langle M\rangle.Q :: z : C}\ \forall L$$

- 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

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

- Left rule: matching use of service

$$\frac{\Psi \vdash M : \tau \quad \Psi \,;\, \Gamma \,;\, \Delta', x{:}A\{M/y\} \Rightarrow Q :: z : C}{\Psi \,;\, \Gamma \,;\, \Delta', x{:}\forall y{:}\tau.A \Rightarrow \overline{x}\langle M\rangle.Q :: z : C}\ \forall L$$

- Proof reduction

$$(\boldsymbol{\nu} x)(x(y).P \mid \overline{x}\langle M\rangle.Q) \longrightarrow (\boldsymbol{\nu} x)(P\{M/y\} \mid Q)$$

Term passing: other connectives

- Quantified proposition as dependent session types

$x : \forall y{:}\tau.A$	Input an $M : A$ along x and behave as $A\{M/y\}$
$x : \$\tau \multimap A$	Input an $M : A$ along x and behave as A
$x : \exists y{:}\tau.A$	Output an $M : A$ along x and behave as $A\{M/y\}$
$x : \$\tau \otimes A$	Output an $M : A$ along x and behave as A

- $\$\tau \multimap A$ 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

$$\begin{array}{lcl} \mathsf{index}_3 & : & !(\mathsf{file} \multimap \mathsf{file} \otimes \mathbf{1}) \\ \mathsf{index}_4 & : & !(\forall f{:}\mathsf{file}.\, \mathsf{pdf}(f) \multimap \exists g{:}\mathsf{file}.\, \mathsf{pdf}(g) \otimes \mathbf{1}) \end{array}$$

  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

$$\begin{array}{lcl} \mathsf{store}_1 & : & !(\mathsf{file} \multimap !(\mathsf{file} \otimes \mathbf{1})) \\ \mathsf{store}_2 & : & !(\forall f{:}\mathsf{file}.\, !\exists g{:}\mathsf{file}.\, g \doteq f \otimes \mathbf{1}) \end{array}$$

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

Outline

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 $g \doteq f$ (by reflexivity) would not be informative
- Use proof irrelevance in type theory
- $M : [\tau]$ — M is a term of type τ that is computationally irrelevant

Proof irrelevance: rules

- Introduction and elimination

$$\frac{\Psi^{\oplus} \vdash M : \tau}{\Psi \vdash [M] : [\tau]}\ []I \qquad \frac{\Psi \vdash M : [\tau] \quad \Psi, x \div \tau \vdash N : \sigma}{\Psi \vdash \mathbf{let}\ [x] = M\ \mathbf{in}\ N : \sigma}\ []E$$

- $\Psi^{\oplus}$ promotes hypotheses $x \div \tau$ to $x{:}\tau$
- In examples, may use pattern matching instead of **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

$$\begin{array}{lcl} \mathsf{index}_4 & : & !(\forall f{:}\mathsf{file}.\, \mathsf{pdf}(f) \multimap \exists g{:}\mathsf{file}.\, \mathsf{pdf}(g) \otimes \mathbf{1}) \\ \mathsf{index}_5 & : & !(\forall f{:}\mathsf{file}.\, [\mathsf{pdf}(f)] \multimap \exists g{:}\mathsf{file}.\, [\mathsf{pdf}(g)] \otimes \mathbf{1}) \end{array}$$

- Persistent file storage

$$\begin{array}{lcl} \mathsf{store}_2 & : & !(\forall f{:}\mathsf{file}.\, !\exists g{:}\mathsf{file}.\, g \doteq f \otimes \mathbf{1}) \\ \mathsf{store}_3 & : & !(\forall f{:}\mathsf{file}.\, !\exists g{:}\mathsf{file}.\, [g \doteq f] \otimes \mathbf{1}) \end{array}$$

- After erasure, communication can be optimized further

Example: mobile code

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

$$\mathsf{index}_5 : !((\Pi f{:}\mathsf{file}.\ [\mathsf{pdf}(f)] \to \Sigma g{:}\mathsf{file}.\ [\mathsf{pdf}(g)]) \otimes \mathbf{1})$$

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

Outline

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 $\Diamond_K \tau$ = 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
  - $[T \to S] = ![T] \multimap [S]$: copying evaluation (by name)
  - $(T \to S)^* = !(T^* \multimap S^*)$: 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 π-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?

Summary

- A Curry-Howard isomorphism
  - Linear propositions as session types
    $A \multimap B$ (input), $A \otimes B$ (output), $\mathbf{1}$ (termination)
    $A \& B$ (external choice), $A \oplus B$ (internal choice), $!A$ (replication)
  - Sequent proofs as π-calculus processes
    with a binary guarded choice and channel forwarding
  - Cut reduction as π-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)
  - $\Diamond_K \tau$ for affirmations (evidenced by digital signatures)