Interconnectability of Session-based Logical Processes

Bernardo Toninho, Nobuko Yoshida

POPL, Lisbon
16 January, 2019
Multiparty Sessions vs Logic-based Sessions

What kind of processes and sessions?

- **Multiparty** Session π-calculus + Multiparty sessions [HYC08,CDYP16]

**Multiparty Session Framework (MP):**

- Types describe interactions between an arbitrary number of participants.
- **Global** Types – Global view of the communication protocol.
- **Local** Types – Participant’s view of the protocol.
Multiparty Sessions vs Logic-based Sessions

What kind of processes and sessions?

- **Multiparty** Session $\pi$-calculus + Multiparty sessions [HYC08,CDYP16]

**Multiparty Session Framework** (MP):
- Types describe interactions between an arbitrary number of participants.
- **Global** Types – Global view of the communication protocol.
- **Local** Types – Participant’s view of the protocol.
- Global types ensure progress within a multiparty session.
Multiparty Sessions vs Logic-based Sessions

What kind of processes and sessions?

- **Multiparty** Session $\pi$-calculus + Multiparty sessions [HYC08, CDYP16]
- **Binary** Session $\pi$-calculus + Logical sessions [CP10, W12, CPT16]

Classical Logical Processes (CLL):

- Session types are linear logic propositions ($A \otimes B$, $A \multimap B$, $A \& B$, etc.).
- Session $\pi$-calculus processes as proof terms of linear logic.
Multiparty Sessions vs Logic-based Sessions

What kind of processes and sessions?

- **Multiparty** Session $\pi$-calculus + Multiparty sessions [HYC08, CDYP16]
- **Binary** Session $\pi$-calculus + Logical sessions [CP10, W12, CPT16]

**Classical Logical Processes** (CLL):

- Session types are linear logic propositions ($A \otimes B$, $A \lla B$, $A \& B$, etc.).
- Session $\pi$-calculus processes as proof terms of linear logic.
- Every process pair shares **exactly** one (compatible) communication channel.
- Strong properties, compositionally (e.g., deadlock-freedom, strong normalization).
Interconnection Networks [AGN95]

\[ p \rightarrow q: (\text{nat}). \quad p \rightarrow r: (\text{bool}). \quad r \rightarrow q: (\text{str}). \quad \text{end} \]
Interconnection Networks [AGN95]

Multiparty network topology:

\[ p \rightarrow q: (\text{nat}). p \rightarrow r: (\text{bool}). r \rightarrow q: (\text{str}). \text{end} \]
Interconnection Networks [AGN95]

Multiparty network topology:

\[ p \rightarrow q:(\text{nat}).p \rightarrow r:(\text{bool}).r \rightarrow q:(\text{str}).\text{end} \]
Multiparty network topology:

\[ p \rightarrow q: (\text{nat}). p \rightarrow r: (\text{bool}). r \rightarrow q: (\text{str}). \text{end} \]
Interconnection Networks [AGN95]

Multiparty network topology:

\[ p \rightarrow q: \text{(nat)}. p \rightarrow r: \text{(bool)}. r \rightarrow q: \text{(str)}. \text{end} \]
Interconnection Networks [AGN95]

Multiparty network topology:

\[ p \rightarrow q: (\text{nat}). p \rightarrow r: (\text{bool}). r \rightarrow q: (\text{str}). \text{end} \]

What happens when we realize this as a logical process network?
Interconnection Networks [AGN95]

Multiparty network topology:

\[ p \rightarrow q:\text{nat}. p \rightarrow r:\text{bool}. r \rightarrow q:\text{str}. \text{end} \]

What happens when we realize this as a logical process network?
Interconnection Networks [AGN95]

Multiparty network topology:

\[ p \rightarrow q :(\text{nat}) . p \rightarrow r :(\text{bool}) . r \rightarrow q :(\text{str}) . \text{end} \]

What happens when we realize this as a logical process network?
Interconnection Networks [AGN95]

Multiparty network topology:

\[ p \to q : \text{(nat)}. \quad p \to r : \text{(bool)}. \quad r \to q : \text{(str)}. \quad \text{end} \]

What happens when we realize this as a logical process network? [CP16,CLMSW16]
Interconnection Networks [AGN95]

Multiparty network topology:

![Network Diagram]

Logical Processes:

![Process Diagrams]

In what precise sense do multiparty networks generalize logical process networks?
Contributions

- **Negative** Result: CLL cannot represent general MP networks.
- **Positive** Result: A single MP session is enough to represent CLL networks.
Contributions

- **Negative** Result: CLL cannot represent general MP networks.
- **Positive** Result: A single MP session is enough to represent CLL networks.
- **Positive** Result: MP can be used to augment CLL networks.
Contributions

- **Negative** Result: CLL cannot represent general MP networks.
- **Positive** Result: A single MP session is enough to represent CLL networks.
- **Positive** Result: MP can be used to augment CLL networks.

A framework of *interconnection networks* in CLL and MP:

- A new way of measuring expressiveness of session processes.
- **Compositional** global type *synthesis* to study and augment CLL networks.
Multiparty Session Processes – MP
By Example

\[ Q \triangleq s[q][p]\langle 7 \rangle; s[q][r](z); s[q][p]\langle "hello" \rangle \]

A global type (synthesis or projection) is needed to ensure deadlock-freedom!

Bernardo Toninho, Nobuko Yoshida
Interconnectability of Session-based Logical Processes
Multiparty Session Processes – MP
By Example

\[ Q \triangleq s[q][p]\langle 7 \rangle; s[q][r](z); s[q][p]\langle "hello" \rangle \]
\[ P \triangleq s[p][q](x); s[p][r]\langle 8 \rangle; s[p][q](y) \]
Multiparty Session Processes – MP
By Example

\[
Q \triangleq s[q][p]⟨7⟩; s[q][r](z); s[q][p]⟨"hello"⟩ \\
P \triangleq s[p][q](x); s[p][r]⟨8⟩; s[p][q](y) \\
R \triangleq s[r][q]⟨1⟩; s[r][p](x)
\]
Multiparty Session Processes – MP
By Example

\[ Q \trianglerighteq s[q][p]\langle 7 \rangle; s[q][r]\langle z \rangle; s[q][p]\langle "hello" \rangle \quad P \trianglerighteq s[p][q]\langle x \rangle; s[p][r]\langle 8 \rangle; s[p][q]\langle y \rangle \]

\[ R \trianglerighteq s[r][q]\langle 1 \rangle; s[r][p]\langle x \rangle \]

\[ Q \vdash_{MP} s[q]: p \uparrow (\text{int}); r \downarrow (\text{int}); p \uparrow (\text{str}); \text{end} \]
Multiparty Session Processes – MP

By Example

\[ Q \triangleq s[q][p]\langle 7 \rangle; s[q][r](z); s[q][p]\langle \text{"hello"} \rangle \]
\[ P \triangleq s[p][q](x); s[p][r]\langle 8 \rangle; s[p][q](y) \]
\[ R \triangleq s[r][q]\langle 1 \rangle; s[r][p](x) \]

\[
\begin{align*}
Q & \vdash_{\text{MP}} s[q]:p \uparrow(\text{int}); r \downarrow(\text{int}); p \uparrow(\text{str}); \text{end} \\
P & \vdash_{\text{MP}} s[p]:q \downarrow(\text{int}); r \uparrow(\text{int}); q \downarrow(\text{str}); \text{end}
\end{align*}
\]
Multiparty Session Processes – MP

By Example

\[ Q \triangleq s[q][p]⟨7⟩; s[q][r](z); s[q][p]⟨"hello"⟩ \]
\[ P \triangleq s[p][q](x); s[p][r]⟨8⟩; s[p][q](y) \]
\[ R \triangleq s[r][q]⟨1⟩; s[r][p](x) \]

\[ Q \vdash_{\text{MP}} s[q]: p^\uparrow(\text{int}); r^\downarrow(\text{int}); p^\uparrow(\text{str}); \text{end} \]
\[ P \vdash_{\text{MP}} s[p]: q^\downarrow(\text{int}); r^\uparrow(\text{int}); q^\downarrow(\text{str}); \text{end} \]
\[ R \vdash_{\text{MP}} s[r]: q^\uparrow(\text{int}); p^\downarrow(\text{int}); \text{end} \]
Multiparty Session Processes – MP

By Example

\[ Q \triangleq s[q][p]⟨7⟩; s[q][r](z); s[q][p]⟨"hello"⟩ \quad P \triangleq s[p][q](x); s[p][r]⟨8⟩; s[p][q](y) \]

\[ R \triangleq s[r][q]⟨1⟩; s[r][p](x) \]

\[ Q \vdash_{MP} s[q]:p \uparrow (\text{int}); r \downarrow (\text{int}); p \uparrow (\text{str}); \text{end} \quad P \vdash_{MP} s[p]:q \downarrow (\text{int}); r \uparrow (\text{int}); q \downarrow (\text{str}); \text{end} \]

\[ R \vdash_{MP} s[r]:q \uparrow (\text{int}); p \downarrow (\text{int}); \text{end} \]

\[ P \mid Q \mid R \vdash_{MP} s[p]:T_1, s[q]:T_2, s[r]:T_3 \]
Multiparty Session Processes – MP
By Example

\[ Q \triangleq s[q][p]\langle 7 \rangle; s[q][r](z); s[q][p]\langle "hello" \rangle \]
\[ P \triangleq s[p][q](x); s[p][r]\langle 8 \rangle; s[p][q](y) \]
\[ R \triangleq s[r][q]\langle 1 \rangle; s[r][p](x) \]

\[ Q \vdash_{\text{MP}} s[q]:p \uparrow(\text{int}); r \downarrow(\text{int}); p \uparrow(\text{str}); \text{end} \]
\[ P \vdash_{\text{MP}} s[p]:q \downarrow(\text{int}); r \uparrow(\text{int}); q \downarrow(\text{str}); \text{end} \]
\[ R \vdash_{\text{MP}} s[r]:q \uparrow(\text{int}); p \downarrow(\text{int}); \text{end} \]

\[(\nu s) (P \mid Q \mid R) \vdash_{\text{MP}} \emptyset\]

A global type (synthesis or projection) is needed to ensure **deadlock-freedom**!
From CLL to MP, and more

Roadmap

1. A type-preserving mapping of CLL channels to MP channels:
   - Treat each **thread** as the implementation of a role
   - Distinct channels link with **distinct** roles
   - Infer a **local** type from the process **actions**
From CLL to MP, and more

Roadmap

1. A type-preserving mapping of CLL channels to MP channels:
   - Treat each thread as the implementation of a role
   - Distinct channels link with distinct roles
   - Infer a local type from the process actions

2. Synthesising global types and interconnection networks, compositionally.
From CLL to MP, and more

Roadmap

1. A type-preserving mapping of CLL channels to MP channels:
   - Treat each thread as the implementation of a role
   - Distinct channels link with distinct roles
   - Infer a local type from the process actions

2. Synthesising global types and interconnection networks, compositionally.

3. Analyse limitations of CLL interconnection networks.

4. Use compositional synthesis to augment CLL composition.
From CLL to MP

- A translation $P \vdash^\sigma_{\text{CL}} \Delta$ that maps CLL channels to role-based MP channels.
- Principle to handle composition (i.e. actions and roles match consistently).
From CLL to MP

- A translation $P \vdash^\sigma_{CL} \Delta$ that maps CLL channels to **role-based** MP channels.
- Principle to handle composition (i.e. actions and roles **match** consistently).
- A **local type** generating function $\llbracket P \rrbracket_\sigma$
From CLL to MP

- A translation $P \vdash^\sigma_{\text{CL}} \Delta$ that maps CLL channels to role-based MP channels.
- Principle to handle composition (i.e. actions and roles match consistently).
- A local type generating function $\llbracket P \rrbracket_\sigma$

Putting it all together:

$$P \triangleq x \langle 7 \rangle . y(z) . x \langle "hello" \rangle$$
From CLL to MP

- A translation $P \vdash_{\text{CLL}}^\sigma \Delta$ that maps CLL channels to role-based MP channels.
- Principle to handle composition (i.e. actions and roles match consistently).
- A local type generating function $\llbracket P \rrbracket_\sigma$

Putting it all together:

$$P \triangleq x\langle 7 \rangle.y(z).x\langle \text{"hello"} \rangle \quad \sigma_1(P) = s[p][q]\langle 7 \rangle; s[p][r](z); s[p][q]\langle \text{"hello"} \rangle$$
From CLL to MP

- A translation $P \vdash^\sigma_{CL} \Delta$ that maps CLL channels to role-based MP channels.
- Principle to handle composition (i.e. actions and roles match consistently).
- A local type generating function $\llbracket P \rrbracket_\sigma$

Putting it all together:

$$P \triangleq x \langle 7 \rangle . y(z). x \langle "hello" \rangle$$

$$\sigma_1 (P) = s[p][q] \langle 7 \rangle ; s[p][r](z) ; s[p][q] \langle "hello" \rangle$$

$$\llbracket P \rrbracket_{\sigma_1} = q \uparrow (\text{int}) ; r \downarrow (\text{int}) ; q \uparrow (\text{str}) ; \text{end}$$
A translation $P \vdash^\sigma_{\text{CL}} \Delta$ that maps CLL channels to role-based MP channels.

Principle to handle composition (i.e. actions and roles match consistently).

A local type generating function $\llbracket P \rrbracket_\sigma$

Putting it all together:

$$P \triangleq x \langle 7 \rangle . y(z) . x \langle "hello" \rangle$$

$$\sigma_1(P) = s[p][q] \langle 7 \rangle ; s[p][r](z) ; s[p][q] \langle "hello" \rangle$$

$$\llbracket P \rrbracket_{\sigma_1} = q \uparrow (\text{int}); r \downarrow (\text{int}); q \uparrow (\text{str}); \text{end}$$

$$R \triangleq y \langle 2 \rangle$$
From CLL to MP

- A translation $P \vdash_{\text{CLL}}^\sigma \Delta$ that maps CLL channels to role-based MP channels.
- Principle to handle composition (i.e. actions and roles match consistently).
- A **local type** generating function $\llbracket P \rrbracket_\sigma$

Putting it all together:

\[
P \triangleq x\langle 7 \rangle.y(z).x\langle \text{"hello"} \rangle
\]

\[
\sigma_1(P) = s[p][q]\langle 7 \rangle; s[p][r](z); s[p][q]\langle \text{"hello"} \rangle
\]

\[
\llbracket P \rrbracket_{\sigma_1} = q \uparrow (\text{int}); r \downarrow (\text{int}); q \uparrow (\text{str}); \text{end}
\]

\[
R \triangleq y\langle 2 \rangle
\]

\[
\sigma_2(R) = s[r][p]\langle 2 \rangle
\]

\[
\llbracket R \rrbracket_{\sigma_2} = p \uparrow (\text{int}); \text{end}
\]
From CLL to MP

- A translation $P \vdash_{CL}^\sigma \Delta$ that maps CLL channels to role-based MP channels.
- Principle to handle composition (i.e. actions and roles match consistently).
- A local type generating function $\llbracket P \rrbracket_\sigma$

Putting it all together:

$$P \triangleq x\langle 7 \rangle.y(z).x\langle "hello" \rangle$$

$$\sigma_1(P) = s[p][q]\langle 7 \rangle; s[p][r](z); s[p][q]\langle "hello" \rangle$$

$$\llbracket P \rrbracket_{\sigma_1} = q\uparrow(int); r\downarrow(int); q\uparrow(str); \text{end}$$

$$R \triangleq y\langle 2 \rangle$$

$$\sigma_2(R) = s[r][p]\langle 2 \rangle$$

$$\llbracket R \rrbracket_{\sigma_2} = p\uparrow(int); \text{end}$$

Bernardo Toninho, Nobuko Yoshida
Interconnectability of Session-based Logical Processes
From CLL to MP

Properties of mapping:

- Type Preservation: $P \vdash_{\text{CL}} \Delta$ implies $\sigma(P) \vdash_{\text{MP}} s[p]:\llbracket P \rrbracket_\sigma$

- Operationally sound and complete.

- Thread Preservation (i.e. a cut-free CLL process maps to a single MP role).

- Uniqueness: All thread-preserving, typable mappings are captured.
From CLL to MP

Properties of mapping:

- Type Preservation: $P \vdash_{\text{CLL}} \Delta$ implies $\sigma(P) \vdash_{\text{MP}} s[p]:[P]_{\sigma}$
- Operationally sound and complete.

Ok but... how do we get global types (and interconnection networks)?
From CLL to MP

Properties of mapping:

- **Type Preservation:** $P \vdash_{\text{CL}} \sigma \Delta$ implies $\sigma(P) \vdash_{\text{MP}} s[p]:[P]_\sigma$
- Operationally sound and complete.
- **Thread Preservation** (i.e. a cut-free CLL process maps to a **single** MP role).
From CLL to MP

Properties of mapping:

- **Type Preservation**: \( P \vdash_{\text{CL}} \Delta \) implies \( \sigma(P) \vdash_{\text{MP}} s[p]:[P]_\sigma \)
- Operationally sound and complete.
- **Thread Preservation** (i.e. a cut-free CLL process maps to a **single** MP role).
- Uniqueness: **All** thread-preserving, typable mappings are captured.
From CLL to MP

Properties of mapping:

▶ Type Preservation: \( P \vdash_{\text{CLL}} \Delta \) implies \( \sigma(P) \vdash_{\text{MP}} s[p]:\llbracket P \rrbracket_\sigma \)

▶ Operationally sound and complete.

▶ **Thread Preservation** (i.e. a cut-free CLL process maps to a **single** MP role).

▶ Uniqueness: **All** thread-preserving, typable mappings are captured.

Ok but... how do we get global types (and interconnection networks)?
Compositional Global Type Synthesis

Intuition:

- Extract a role’s contribution to a global session from the process/local type (i.e., a partial global type).
- Combine, or fuse, partial global types (only defined when compatible).
Compositional Global Type Synthesis

Intuition:
- Extract a role’s contribution to a global session from the process/local type (i.e., a partial global type).
- Combine, or fuse, partial global types (only defined when compatible).

\[(s[p]:q \uparrow\text{int}; r \downarrow\text{int}; q \uparrow\text{str}) = p \rightsquigarrow q: \uparrow\text{int}. \ r \rightsquigarrow p: \downarrow\text{int}. \ p \rightsquigarrow q: \uparrow\text{str} \quad (G_1)\]
Compositional Global Type Synthesis

Intuition:
- Extract a role’s contribution to a global session from the process/local type (i.e., a partial global type).
- Combine, or fuse, partial global types (only defined when compatible).

\[
(s[p]:q \uparrow \text{(int)}; r \downarrow \text{(int)}; q \uparrow \text{(str)}) = p \rightsquigarrow q: \uparrow \text{(int)}. \quad r \rightsquigarrow p: \downarrow \text{(int)}. \quad p \rightsquigarrow q: \uparrow \text{(str)} \quad (G_1)
\]

\[
(s[r]:p \uparrow \text{(int)}) = \quad r \rightsquigarrow p: \uparrow \text{(int)} \quad (G_2)
\]
Compositional Global Type Synthesis

Intuition:

- Extract a role's contribution to a global session from the process/local type (i.e., a partial global type).
- Combine, or fuse, partial global types (only defined when compatible).

\[
|s[p]:q \uparrow \text{(int)}; r \downarrow \text{(int)}; q \uparrow \text{(str)}| = p \leftrightarrow q: \uparrow \text{(int)}. \ r \leftrightarrow p: \downarrow \text{(int)}. \ p \leftrightarrow q: \uparrow \text{(str)} \quad (G_1)
\]
\[
|s[r]:p \uparrow \text{(int)}| = r \leftrightarrow p: \uparrow \text{(int)} \quad (G_2)
\]

\[
\text{fuse}(G_1, G_2) = p \leftrightarrow q: \uparrow \text{(int)}. \ r \rightarrow p: \text{(int)}. \ p \rightarrow q: \uparrow \text{(str)}
\]

A global type induces an interconnection network graph in the expected way.
Compositional Global Type Synthesis
Putting it all together

Given \( P \vdash_{\text{CL}} \Delta \):
- Map into MP: \( P \vdash_{\sigma} \Delta \)
Compositional Global Type Synthesis
Putting it all together

Given $P \vdash_{\text{cl}} \Delta$:

- Map into MP: $P \vdash_{\text{cl}} \Delta$
- Infer, or synthesise, **local** and **partial** types: $\llbracket P \rrbracket_\sigma, (\rightarrow)_\sigma$
- Compose processes (and mappings) consistently: $P \vdash_{\rho} \Delta; \Gamma$

Key Results: Let $P \vdash_{\rho} \Delta; \Gamma$,

- **Fusion**: If $P$ is closed, then $\llbracket P \rrbracket_\sigma$ is a unique and complete global type.
- **No Cycles**: The interconnection network induced by $\llbracket P \rrbracket_\sigma$ is acyclic.
Compositional Global Type Synthesis
Putting it all together

Given $P \vdash_{cl} \Delta$:

- Map into MP: $P \vdash_{cl}^{\sigma} \Delta$
- Infer, or synthesise, local and partial types: $\llbracket P \rrbracket^{\sigma}$, $(\neg)_{\rho}^{\sigma}$
- Compose processes (and mappings) consistently: $P \vdash_{\rho}^{\sigma} \Delta; \Gamma$
- Compose partial types consistently: fuse($\llbracket \Gamma \rrbracket^{\sigma}$)
Given $P \vdash_{\text{cl}} \Delta$:

- Map into MP: $P \vdash_{\text{cl}}^{\sigma} \Delta$
- Infer, or synthesise, local and partial types: $\llbracket P \rrbracket^{\sigma}, (\lnot)_{\sigma}$
- Compose processes (and mappings) consistently: $P \vdash_{\rho}^{\sigma} \Delta; \Gamma$
- Compose partial types consistently: $\text{fuse}(\llbracket \Gamma \rrbracket^{\sigma})$

Key Results: Let $P \vdash_{\rho}^{\sigma} \Delta; \Gamma$,

- **Fusion**: If $P$ is closed, then $\llbracket \Gamma \rrbracket^{\sigma}$ is a unique and complete global type.
Compositional Global Type Synthesis
Putting it all together

Given $P \vdash_{CL} \Delta$:

- Map into $MP$: $P \vdash_{CL}^\sigma \Delta$
- Infer, or synthesise, local and partial types: $\llbracket P \rrbracket^\sigma, (\neg)\sigma$
- Compose processes (and mappings) consistently: $P \vdash^\sigma \rho \Delta; \Gamma$
- Compose partial types consistently: fuse($\llbracket \Gamma \rrbracket^\sigma$)

Key Results: Let $P \vdash^\sigma \rho \Delta; \Gamma$,

- **Fusion**: If $P$ is closed, then $\llbracket \Gamma \rrbracket^\sigma$ is a unique and complete global type.
- **No Cycles**: The interconnection network induced by $\llbracket \Gamma \rrbracket^\sigma$ is acyclic.
Compositional Global Type Synthesis

Key Results: Let $P \vdash^\text{σ} \Delta; \Gamma$, 

- **Fusion**: If $P$ is closed, then $(|\Gamma|)^\text{σ}$ is a **unique** and **complete** global type.
  - **Positive** Result: A single MP session is enough to represent CLL networks.
Compositional Global Type Synthesis

Key Results: Let $P \vdash^\sigma \Delta; \Gamma$,

- **Fusion**: If $P$ is closed, then $(\Gamma)^\sigma$ is a unique and complete global type.
  - **Positive** Result: A single MP session is enough to represent CLL networks.
- **No Cycles**: The interconnection network induced by $(\Gamma)^\sigma$ is acyclic.
  - **Negative** Result: CLL cannot represent general MP networks.
- Results scale to delegation and replication in CLL.
Cycles can deadlock (e.g. $x\langle 1 \rangle . y\langle 2 \rangle | y(z).x(w)$). Disallowed by CLL composition:

\[
\begin{array}{l}
\text{(cut)} \quad P \vdash_{\text{CL}} \Delta, \ x : A \quad Q \vdash_{\text{CL}} \Delta', \ x : A^\perp \\
\frac{}{(\nu x)(P | Q) \vdash_{\text{CL}} \Delta, \Delta'}
\end{array}
\]

Fuseable global types are safe, so cannot result in deadlocks!
Beyond CLL Networks

- Cycles **can** deadlock (e.g. \( x\langle 1 \rangle . y\langle 2 \rangle \mid y(z). x(w) \)). Disallowed by CLL composition:

\[
\frac{P \vdash_{\text{CL}} \Delta, x:A \quad Q \vdash_{\text{CL}} \Delta', x:A^\perp}{(\nu x)(P \mid Q) \vdash_{\text{CL}} \Delta, \Delta'}
\]

- Fuseable global types are **safe**, so cannot result in deadlocks!
Beyond CLL Networks

- Cycles can deadlock (e.g. \( x \langle 1 \rangle . y \langle 2 \rangle | y(z).x(w) \)). Disallowed by CLL composition:

\[
\begin{align*}
\text{(cut)} & \quad P \vdash_{\text{cl}} \Delta, x:A & Q \vdash_{\text{cl}} \Delta', x:A^\perp \\
& \quad \therefore (\nu x)(P | Q) \vdash_{\text{cl}} \Delta, \Delta'
\end{align*}
\]

- Fuseable global types are safe, so cannot result in deadlocks!

\[
\begin{align*}
P \vdash^{\sigma_1} \Delta, x_1:A_1, \ldots, x_n:A_n & \quad Q \vdash^{\sigma_2} \Delta', x_1:A_1^\perp, \ldots, x_n:A_n^\perp
\end{align*}
\]
Beyond CLL Networks

Cycles can deadlock (e.g. \(x\langle 1\rangle.y\langle 2\rangle \mid y(z).x(w)\)). Disallowed by CLL composition:

\[
\frac{P \vdash_{\text{cl}} \Delta, x:A \quad Q \vdash_{\text{cl}} \Delta', x:A^\perp}{(\nu x)(P \mid Q) \vdash_{\text{cl}} \Delta, \Delta'}
\]

Fuseable global types are safe, so cannot result in deadlocks!
Beyond CLL Networks

- Cycles can deadlock (e.g. \(x(1).y(2) | y(z).x(w)\)). Disallowed by CLL composition:

\[
\begin{align*}
\text{(cut)} & \quad \frac{P \vdash_{\text{CL}} \Delta, x:A \quad Q \vdash_{\text{CL}} \Delta', x:A^\perp}{(\nu x)(P | Q) \vdash_{\text{CL}} \Delta, \Delta'}
\end{align*}
\]

- Fuseable global types are safe, so cannot result in deadlocks!

\[
\begin{align*}
P \vdash^{\sigma_1} \Delta, x_1:A_1, \ldots, x_n:A_n & \quad Q \vdash^{\sigma_2} \Delta', x_1:A_1^\perp, \ldots, x_n:A_n^\perp \\
\text{fuse}(G_1, G_2) \text{ defined} & \quad (\nu x_1, \ldots, x_n)(P | Q) \vdash^{\sigma_3} \Delta, \Delta'
\end{align*}
\]
Concluding Remarks

- A new framework for measuring expressiveness of various session processes.
- A precise argument for the added expressiveness of MP over logical sessions.
- MP can inform and extend CLL composition with cyclic, deadlock-free networks.
- CLL composition can inform and extend MP deadlock-free session interleaving.
- Other approaches:
  - Full abstraction/inverse between λ-calculus and logical sessions [TY18]
  - Beyond logic (e.g., sharing [BP17, BPT18], effects [OY16], circularity [DG18]).

Thank you for your time!

Questions?
Concluding Remarks

- A new framework for measuring expressiveness of various session processes.
- A precise argument for the added expressiveness of MP over logical sessions.
- MP can inform and extend CLL composition with cyclic, deadlock-free networks.
- CLL composition can inform and extend MP deadlock-free session interleaving.
- Other approaches:
  - Full abstraction/inverse between λ-calculus and logical sessions [TY18]
  - Beyond logic (e.g., sharing [BP17,BPT18], effects [OY16], circularity [DG18]).

Thank you for your time!

Questions?