Duality of Session Types: The Final Cut

Simon J. Gay  Peter Thiemann  Vasco T. Vasconcelos

University of Glasgow  University of Freiburg  University of Lisbon

VEST
online, June 2020
Session Types — Types for Structured Communication

\[ S ::= \ ! T . S' \quad \text{send} \]
\[ ? T . S' \quad \text{receive} \]
Session Types — Types for Structured Communication

\[
S ::= !T.S' \quad \text{send} \\
?T.S' \quad \text{receive} \\
\oplus \{ \ell_i : S_i \} \quad \text{select} \\
\& \{ \ell_i : S_i \} \quad \text{choice}
\]
Session Types — Types for Structured Communication

\[ S ::= !T.S' \quad \text{send} \]
\[ ?T.S' \quad \text{receive} \]
\[ \oplus \{ l_i : S_i \} \quad \text{select} \]
\[ \& \{ l_i : S_i \} \quad \text{choice} \]
\[ \text{end} \]
The good old math server

Session type of the server

type Server = &{
    Neg: ?Int. !Int. end,
    Add: ?Int. ?Int. !Int. end}

Session type of the client

type Client = ⊕{
    Neg: !Int. ?Int. end,
    Add: !Int. ?Int. !Int. end}

Client type is the dual of server type
The good old math server

Session type of the server

type Server = &{
    Neg: ?Int. !Int. end,
    Add: ?Int. ?Int. !Int. end}

Session type of the client

type Client = ⊕{
    Neg: !Int. ?Int. end,
    Add: !Int. !Int. ?Int. end}
The good old math server

Session type of the server

```plaintext
type Server = &{
    Neg: ?Int. !Int. end,
    Add: ?Int. ?Int. !Int. end
}
```

Session type of the client

```plaintext
type Client = ⊕{
    Neg: !Int. ?Int. end,
    Add: !Int. !Int. ?Int. end
}
```

Client type is the dual of server type
Duality

Definition 1

\[
\bar{\text{end}} = \text{end} \quad \bar{!T.S} = ?T.S \quad \bar{\oplus}\{\ell_i : S_i\} = \&\{\ell_i : \bar{S}_i\} \\
\bar{?T.S} = !T.S \quad \bar{\&}\{\ell_i : S_i\} = \bar{\oplus}\{\ell_i : \bar{S}_i\}
\]
Duality

Definition 1

\[ \text{end} = \text{end} \quad \overline{!T.S} = ?T.\overline{S} \quad \overline{\oplus \{\ell_i : S_i\}} = \& \{\ell_i : \overline{S_i}\} \]
\[ \overline{?T.S} = !T.\overline{S} \quad \overline{\& \{\ell_i : S_i\}} = \oplus \{\ell_i : \overline{S_i}\} \]

Undebatably correct!

▶ Kohei Honda (CONCUR 1993): Types for Dyadic Interaction.
▶ Kaku Takeuchi, Kohei Honda, Makoto Kubo (PARLE1994): An Interaction-based Language and its Typing System.
▶ Kohei Honda, Vasco Thudichum Vasconcelos, Makoto Kubo (ESOP 1998): Language Primitives and Type Discipline for Structured Communication-Based Programming.
Adding Recursion

\[ S ::= \ldots \]

\[ \mu X. S \quad \text{recursive session} \]

\[ X \quad \text{type variable} \]
A more interesting math server

Session type of the server

```
type Server = \mu X. \&\{
  Neg: ?Int. !Int. X,
  Add: ?Int. ?Int. !Int. X,
  Quit: end
\}
```
A more interesting math server

Session type of the server

\[
\text{type Server} = \mu X. \ \{ \\
\quad \text{Neg: } ?\text{Int} . \ !\text{Int} . \ X , \\
\quad \text{Add: } ?\text{Int} . \ ?\text{Int} . \ !\text{Int} . \ X , \\
\quad \text{Quit: end} \} 
\]

Session type of the client

\[
\text{type Client} = \mu X. \ \{ \\
\quad \text{Neg: } !\text{Int} . \ ?\text{Int} . \ X , \\
\quad \text{Add: } !\text{Int} . \ !\text{Int} . \ ?\text{Int} . \ X , \\
\quad \text{Quit: end} \} 
\]
Naive Duality

Definition (extends Definition 1)

\[ \overline{X} = X \]
\[ \overline{\mu X. S} = \mu X. \overline{S} \]
Drawback: Naive Duality is not always correct

Consider

\[ S = \mu X. !X.X \quad = \quad !(\mu X. !X.X)(\mu X. !X.X) \]
Drawback: Naive Duality is not always correct

Consider

\[ S = \mu X. !X.X = !((\mu X. !X.X)(\mu X. !X.X)) = !S.S \]

Unfolding shows that this server wants to send a channel of type \( S \).
Drawback: Naive Duality is not always correct

Consider

\[ S = \mu X. !X.X \quad = \quad !(\mu X. !X.X).(\mu X. !X.X) \quad = \quad !S.S \]

Unfolding shows that this server wants to send a channel of type \( S \). But its naive dual is

\[ \overline{S} = \mu X. !X.X \quad = \quad \mu X. !X.X \quad = \quad \mu X. ?X.X \quad = \quad ?\overline{S}.\overline{S} \]

so the client wrongly expects to receive a channel of type \( \overline{S} \neq S \)!
Goal

Find a satisfactory definition of duality for recursive session types in $\mu$ notation.
Outline

A Coinductive Definition

Bernardi and Hennessy

Lindley and Morris

Mechanization
Recursive Session Types, Coinductively

A recursive type is a potentially infinite tree, labeled by the type constructors. The sets \( \text{Type} \) of type trees and \( \text{SType} \) of session type trees are given by the greatest fixpoint of

\[
F(S, T) = (\{\text{end}\} \cup \{?T.S, !T.S \mid T \in T, S \in S\}) \\
\times (\{\text{int}\} \cup S)
\]

Duality is a binary relation on \( \text{SType} \) defined as the greatest fixpoint of

\[
F(\mathcal{D}) = \{(\text{end}, \text{end})\} \\
\cup \{(?T.S_1, !T.S_2) \mid T \in \text{Type}, (S_1, S_2) \in \mathcal{D}\} \\
\cup \{(!T.S_1, ?T.S_2) \mid T \in \text{Type}, (S_1, S_2) \in \mathcal{D}\}
\]

This gets more involved with the \( \mu \) notation...
Example

\[ \mu X. ?T. X \approx \mu X. ?T. ?T. X \]
\[ \mu X. ?T. X \perp \mu X. !T. !T. X \]
Definition (Syntactic Duality of Session Types)

If \( \mathcal{D} \) is a relation on SType then \( F_\perp(\mathcal{D}) \) is the relation on SType defined by:

\[
F_\perp(\mathcal{D}) = \{(\text{end}, \text{end})\} \\
\cup \{(?T_1.S_1, !T_2.S_2) \mid T_1 \approx T_2 \text{ and } (S_1, S_2) \in \mathcal{D}\} \\
\cup \{(!T_1.S_1, ?T_2.S_2) \mid T_1 \approx T_2 \text{ and } (S_1, S_2) \in \mathcal{D}\} \\
\cup \{(S_1, \mu X.S_2) \mid (S_1, S_2[\mu X.S_2/X]) \in \mathcal{D}\} \\
\cup \{((\mu X.S_1), S_2) \mid (S_1[\mu X.S_1/X], S_2) \in \mathcal{D} \text{ and } S_2 \neq \mu Y.S_3\}
\]

A relation \( \mathcal{D} \) on SType is a session duality if \( \mathcal{D} \subseteq F_\perp(\mathcal{D}) \).

Duality of session types, \( \cdot \perp \cdot \), is the largest session duality.
Outline

A Coinductive Definition

Bernardi and Hennessy

Lindley and Morris

Mechanization
Bernardi and Hennessy’s Discovery (CONCUR 2014)

- Given $S = \mu X. ?X.X$

- But is this correct?

- Let’s rewrite $S$ by unrolling one occurrence of $X$

  $S' = \mu X. \mu X. ?X.X$

- But unrolling yields $S \approx S'$ but $S \not\approx S'$

- Now $S \approx S'$ but $S \not\approx S'$
Given $S = \mu X. ?X.X$

Then $\overline{S} = \mu X. !X.X$ is its naive dual (with $S \not\approx \overline{S}$)
Bernardi and Hennessy's Discovery (CONCUR 2014)

- Given $S = \mu X. ?X.X$
- Then $\overline{S} = \mu X. !X.X$ is its naive dual (with $S \not\approx \overline{S}$)
- But is this correct?
Given $S = \mu X. ?X.X$

Then $\overline{S} = \mu X. !X.X$ is its naive dual (with $S \not\cong \overline{S}$)

But is this correct?

Let’s rewrite $S$ by unrolling one occurrence of $X$
Given \( S = \mu X.\ ?X.X \)

Then \( \overline{S} = \mu X.\ !X.X \) is its naive dual (with \( S \not\approx \overline{S} \))

But is this correct?

Let’s rewrite \( S \) by unrolling one occurrence of \( X \)

\( S' = \mu X.\ ?S.X = \mu X.\ ?(\mu X.\ ?X.X).X \)
Bernardi and Hennessy’s Discovery (CONCUR 2014)

- Given $S = \mu X. ?X.X$
- Then $\overline{S} = \mu X. !X.X$ is its naive dual (with $S \not\approx \overline{S}$)
- But is this correct?
- Let’s rewrite $S$ by unrolling one occurrence of $X$
  - $S' = \mu X. ?S.X = \mu X.?(\mu X. ?X.X).X$
  - $\overline{S'} = \mu X. !S.X$
Bernardi and Hennessy’s Discovery (CONCUR 2014)

- Given $S = \mu X. ?X. X$
- Then $\overline{S} = \mu X. !X. X$ is its naive dual (with $S \not\approx \overline{S}$)
- But is this correct?
- Let’s rewrite $S$ by unrolling one occurrence of $X$
- $S' = \mu X. ?S.X = \mu X. ?(\mu X. ?X.X).X$
- $\overline{S'} = \mu X. !S.X$
- But unrolling yields $\overline{S} \approx \mu X. !\overline{S}.X$
Given $S = \mu X. ?X.X$

Then $\overline{S} = \mu X. !X.X$ is its naive dual (with $S \not\approx \overline{S}$)

But is this correct?

Let’s rewrite $S$ by unrolling one occurrence of $X$

$S' = \mu X. ?S.X = \mu X.?(\mu X. ?X.X).X$

$\overline{S'} = \mu X. !S.X$

But unrolling yields $\overline{S} \approx \mu X. !\overline{S}.X$

Now $S \approx S'$ but $\overline{S} \not\approx \overline{S'}$!
Bernardi and Hennessy’s Solution

BH Duality

- Compute the *message closure* of a session type.
- Apply naive duality to the result.

Definition

A session type is *message-closed* if all message types are closed.

For example

- $S = \mu X. ? X X$ is not message-closed
- $S' = \mu X. ? S X$ is message-closed
Bernardi and Hennessy’s Solution

BH Duality

- Compute the *message closure* of a session type.
- Apply naive duality to the result.

Definition
A session type is *message-closed* if all message types are closed.
Bernardi and Hennessy’s Solution

BH Duality

- Compute the *message closure* of a session type.
- Apply naive duality to the result.

Definition
A session type is *message-closed* if all message types are closed.

For example
- \( S = \mu X.\ ?X.X \) is not message-closed
- \( S' = \mu X.\ ?S.X \) is message-closed
Bernardi and Hennessy’s Results

- BH duality is sound \( \cdot \perp \cdot \).
- but the definition of message closure is quite involved and may increase the size of a type substantially

**Definition (Message Closure [BH2014])**
For any type \( T \) and substitution \( \sigma \) closing for \( T \), the type \( \text{mclo}(T, \sigma) \) is defined inductively by the following rules.

\[
\begin{align*}
\text{mclo}(\text{end}, \sigma) &= \text{end} & \text{mclo}(\text{?}T.S, \sigma) &= \text{?}(T\sigma).\text{mclo}(S, \sigma) \\
\text{mclo}(\text{int}, \sigma) &= \text{int} & \text{mclo}(\text{!}T.S, \sigma) &= \text{!}(T\sigma).\text{mclo}(S, \sigma) \\
\text{mclo}(X, \sigma) &= X & \text{mclo}(\mu X.S, \sigma) &= \mu X.\text{mclo}(S, [(\mu X.S)/X]; \sigma)
\end{align*}
\]

Define \( \text{mclo}(S) \) as \( \text{mclo}(S, \varepsilon) \).
GTV’s optimization

- BH duality can be simplified by symbolic composition of message closure and naive duality (and deforestation)

**Definition (Duality with On-the-fly Message Closure)**

For any session type $S$ and substitution $\sigma$ closing for $S$, the session type $\text{dualof}(S, \sigma)$ is defined inductively by the following rules.

\[
\begin{align*}
\text{dualof}(\text{end}, \sigma) &= \text{end} \\
\text{dualof}(\ ? T . S, \sigma) &= ! (T \sigma) . \text{dualof}(S, \sigma) \\
\text{dualof}(\ ! T . S, \sigma) &= ? (T \sigma) . \text{dualof}(S, \sigma) \\
\text{dualof}(X, \sigma) &= X \\
\text{dualof}(\ \mu X . S, \sigma) &= \mu X . \text{dualof}(S, [\mu X . S / X]; \sigma)
\end{align*}
\]

Define $\text{dualof}(S)$ as $\text{dualof}(S, \varepsilon)$. 
Outline

A Coinductive Definition

Bernardi and Hennessy

Lindley and Morris

Mechanization
Lindley and Morris’s Approach

- Lindley and Morris [ICFP 2016] give another definition of duality

- Each type variable $X$ comes with its companion negative type variable $\neg X$.

- A negative variable $\neg X$ behaves like a suspended application of duality, which gets triggered by substitution for $X$. 
Lindley and Morris’s Approach

- Lindley and Morris [ICFP 2016] give another definition of duality
- It relies on a technical twist, negative type variables, ...
Lindley and Morris’s Approach

- Lindley and Morris [ICFP 2016] give another definition of duality.
- It relies on a technical twist, negative type variables, ...
- Each type variable $X$ comes with its companion *negative type variable* $\overline{X}$. 
Lindley and Morris’s Approach

- Lindley and Morris [ICFP 2016] give another definition of duality.
- It relies on a technical twist, negative type variables, . . .
- Each type variable $X$ comes with its companion *negative type variable* $\overline{X}$
- A negative variable $\overline{X}$ behaves like a suspended application of duality, which gets triggered by substitution for $X$. 
Lindley and Morris’s Solution

Definition (Lindley-Morris Duality, Original Version [ICFP2016])

\[
\begin{align*}
\text{lmd}(\text{end}) & = \text{end} & \text{lmd}(X) & = \overline{X} \\
\text{lmd}(? T . S) & = ! T . \text{lmd}(S) & \text{lmd}(\overline{X}) & = X \\
\text{lmd}(! T . S) & = ? T . \text{lmd}(S) & \text{lmd}(\mu X . S) & = \mu X . (\text{lmd}(S)\{\overline{X}/X\})
\end{align*}
\]

Caveat
▶ The operation \(T\{X/X\}\) is not standard substitution.
▶ It rather swaps \(X\) and \(X\).
Lindley and Morris’s Solution

Definition (Lindley-Morris Duality, Original Version [ICFP2016])

\[ \text{lmd}(\text{end}) = \text{end} \quad \text{lmd}(X) = \overline{X} \]
\[ \text{lmd}(?T.S) = !T.\text{lmd}(S) \quad \text{lmd}(\overline{X}) = X \]
\[ \text{lmd}(!T.S) = ?T.\text{lmd}(S) \quad \text{lmd}(\mu X.S) = \mu X.(\text{lmd}(S)\{\overline{X}/X\}) \]

Caveat

- The operation \( T\{\overline{X}/X\} \) is not standard substitution.
- It rather swaps \( X \) and \( \overline{X} \).

Example

\[ \text{lmd} (\mu X.?X.X) = \mu X.\text{lmd} (?X.X)\{\overline{X}/X\} \]
\[ = \mu X.(!X.\overline{X})\{\overline{X}/X\} \]
\[ = \mu X.(!\overline{X}.X) \]
GTV’s Results on LM Duality

- Unfortunately, Lindley and Morris just state this definition without proof.
GTV’s Results on LM Duality

- Unfortunately, Lindley and Morris just state this definition without proof.
- We prove its soundness in several ways.
GTV’s Results on LM Duality

- Unfortunately, Lindley and Morris just state this definition without proof.
- We prove its soundness in several ways.
  - manually

[https://github.com/peterthiemann/dual-session](https://github.com/peterthiemann/dual-session)
GTV's Results on LM Duality

- Unfortunately, Lindley and Morris just state this definition without proof.
- We prove its soundness in several ways.
  - manually
  - mechanized in Agda
    https://github.com/peterthiemann/dual-session
GTV’s Results on LM Duality

- Unfortunately, Lindley and Morris just state this definition without proof.
- We prove its soundness in several ways.
  - manually
  - mechanized in Agda
    https://github.com/peterthiemann/dual-session
- We observe that it is size-preserving.
Outline

A Coinductive Definition

Bernardi and Hennessy

Lindley and Morris

Mechanization
Some Glimpses at the Agda Code

- Baseline: coinductive definitions of
  - session types with recursion
  - functional and relational duality
- inductive definition of session types with recursion
- definition of LM duality
- correspondence of LM duality with functional duality (new result)
- Not shown:
  - soundness of naive duality for tail recursive session types (new result)
  - definition of BH duality and its soundness
  - what if recursive types are not normalized? contractiveness . . .
- Details in paper at the PLACES 2020 workshop
  https://arxiv.org/abs/2004.01322v1
Plan of proof: soundness of \( \text{Imd} \)

Diagram:

\[
\begin{array}{c}
\text{IND.STType} \quad \xrightarrow{\text{unravel}} \quad \text{COI.SType} \\
\downarrow \quad \quad \quad \quad \downarrow \text{COI.dual} \\
\text{IND.\text{Imdual}} \quad \quad \quad \text{COI.SType} \\
\downarrow \quad \quad \quad \quad \quad \quad \downarrow \sim \\
\text{IND.SType} \quad \xrightarrow{\text{unravel}} \quad \text{COI.SType}
\end{array}
\]
Plan of proof: soundness of naive dual for tail-recursive session types

IND.stail \xrightarrow{\text{unravel}} COI.stype

IND.naive-dual \xrightarrow{\text{unravel}} COI.stype

COI.stype \cong COI.stype
Plan of proof: soundness of message closure

\[
\begin{array}{c}
\text{IND.} \text{SType} \xrightarrow{\text{unravel}} \text{COI.} \text{SType} \\
\text{mclo} \\
\text{IND.} \text{STail} \xrightarrow{\text{unravel}} \text{COI.} \text{SType}
\end{array}
\]
Plan of proof: soundness of message closure

\[
\begin{align*}
\text{IND.SType} & \xrightarrow{\text{unravel}} \text{COI.SType} \\
\text{IND.STail} & \xrightarrow{\text{unravel}} \text{COI.SType} \\
\text{IND.naive-dual} & \xrightarrow{\text{unravel}} \text{COI.dual} \\
\text{IND.STail} & \xrightarrow{\text{unravel}} \text{COI.SType}
\end{align*}
\]
Thank you!