Duality of Session Types: The Final Cut

Simon J. Gay **Peter Thiemann** Vasco T. Vasconcelos

University of Glasgow University of Freiburg University of Lisbon

WG2.8
Zion, March 2020

Session Types — Types for Structured Communication

$$
\begin{array}{lll}
S & ::= & !\,T.S' \qquad \text{send} \\
  &   & ?\,T.S' \qquad \text{receive}
\end{array}
$$

Session Types — Types for Structured Communication

$$
\begin{array}{lcll}
S & ::= & !T.S' & \text{send} \\
 & & ?T.S' & \text{receive} \\
 & & \oplus\{\ell_i : S_i\} & \text{select} \\
 & & \&\{\ell_i : S_i\} & \text{choice}
\end{array}
$$

Session Types — Types for Structured Communication

$$
\begin{array}{llll}
S & ::= & !T.S' & \text{send} \\
  &   & ?T.S' & \text{receive} \\
  &   & \oplus\{\ell_i : S_i\} & \text{select} \\
  &   & \&\{\ell_i : S_i\} & \text{choice} \\
  &   & \mathsf{end} &
\end{array}
$$

The good old math server

Session type of the server

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

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

Duality

Definition

$$\overline{\mathsf{end}} = \mathsf{end} \qquad \overline{!T.S} = ?T.\overline{S} \qquad \overline{\oplus\{\ell_i : S_i\}} = \&\{\ell_i : \overline{S_i}\}$$

$$\overline{?T.S} = !T.\overline{S} \qquad \overline{\&\{\ell_i : S_i\}} = \oplus\{\ell_i : \overline{S_i}\}$$

Duality

Definition

$$\overline{\mathsf{end}} = \mathsf{end} \qquad \overline{!T.S} = ?T.\overline{S} \qquad \overline{\oplus\{\ell_i : S_i\}} = \&\{\ell_i : \overline{S_i}\}$$

$$\overline{?T.S} = !T.\overline{S} \qquad \overline{\&\{\ell_i : S_i\}} = \oplus\{\ell_i : \overline{S_i}\}$$

Undebatably right!

- 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	$::=$	$\ldots$	
		$\mu X.S$	recursive session
		X	type variable

A more interesting math server

Session type of the server

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

A more interesting math server

Session type of the server

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

Session type of the client

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

Naive Duality

Definition

$$\overline{X} = X \qquad \overline{\mu X.S} = \mu X.\overline{S}$$

Naive Duality

Definition

$$\overline{X} = X \qquad \overline{\mu X.S} = \mu X.\overline{S}$$

Only good for tail-recursive session types

- Kohei Honda, Vasco Thudichum Vasconcelos, Makoto Kubo (ESOP 1998): Language Primitives and Type Discipline for Structured Communication-Based Programming.

Outline

Ground Truth

A coinductive definition

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:

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

A relation $\mathcal{D}$ on SType is a *session duality* if $\mathcal{D} \subseteq F_\perp(\mathcal{D})$. Duality of session types, $\perp$, is the largest session duality.

Outline

Bernardi and Hennessy's Discovery (CONCUR 2014)

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

Bernardi and Hennessy's Discovery (CONCUR 2014)

- Given $S = \mu X.?X.X$
- Then $\overline{S} = \mu X.!X.X$

Bernardi and Hennessy's Discovery (CONCUR 2014)

- Given $S = \mu X.?X.X$
- Then $\overline{S} = \mu X.!X.X$
- But is this correct?

Bernardi and Hennessy's Discovery (CONCUR 2014)

- Given $S = \mu X.?X.X$
- Then $\overline{S} = \mu X.!X.X$
- But is this correct?
- Let's rewrite S by unrolling one occurrence of X

Bernardi and Hennessy's Discovery (CONCUR 2014)

- Given $S = \mu X.?X.X$
- Then $\overline{S} = \mu X.!X.X$
- But is this correct?
- Let's rewrite S by unrolling one occurrence of X
- $S' = \mu X.?S.X$

Bernardi and Hennessy's Discovery (CONCUR 2014)

- Given $S = \mu X.?X.X$
- Then $\overline{S} = \mu X.!X.X$
- But is this correct?
- Let's rewrite S by unrolling one occurrence of X
- $S' = \mu X.?S.X$
- $\overline{S'} = \mu X.!S.X$

Bernardi and Hennessy's Discovery (CONCUR 2014)

- Given $S = \mu X.?X.X$
- Then $\overline{S} = \mu X.!X.X$
- But is this correct?
- Let's rewrite S by unrolling one occurrence of X
- $S' = \mu X.?S.X$
- $\overline{S'} = \mu X.!S.X$
- Now $S \sim S'$ but $\overline{S} \not\sim \overline{S'}$!

Bernardi and Hennessy's Solution

BH Duality

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

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 wrt $\perp$
- 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 σ closing for T, the type $\mathsf{mclo}(T, \sigma)$ is defined inductively by the following rules.

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

Define $\mathsf{mclo}(S)$ as $\mathsf{mclo}(S, \varepsilon)$.

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 σ closing for S, the session type $\mathsf{dualof}(S, \sigma)$ is defined inductively by the following rules.

$$
\begin{aligned}
\mathsf{dualof}(\mathsf{end}, \sigma) &= \mathsf{end} & \mathsf{dualof}(?T.S, \sigma) &= !(T\sigma).\,\mathsf{dualof}(S, \sigma) \\
& & \mathsf{dualof}(!T.S, \sigma) &= ?(T\sigma).\,\mathsf{dualof}(S, \sigma) \\
\mathsf{dualof}(X, \sigma) &= X & \mathsf{dualof}(\mu X.S, \sigma) &= \mu X.\mathsf{dualof}(S, [\mu X.S/X]; \sigma)
\end{aligned}
$$

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

Outline

Lindley and Morris's Approach

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

Lindley and Morris's Approach

- Lindley and Morris [ICFP 2016] give another definition of duality
- But it relies on negative type variables ...

Lindley and Morris's Approach

- Lindley and Morris [ICFP 2016] give another definition of duality
- But it relies on 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
- But it relies on 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])

$$\begin{aligned} \mathsf{f}(\mathsf{end}) &= \mathsf{end} & \mathsf{f}(X) &= \overline{X} \\ \mathsf{f}(?T.S) &= !T.\mathsf{f}(S) & \mathsf{f}(\overline{X}) &= X \\ \mathsf{f}(!T.S) &= ?T.\mathsf{f}(S) & \mathsf{f}(\mu X.S) &= \mu X.(\mathsf{f}(S)\{\overline{X}/X\}) \end{aligned}$$

Lindley and Morris's Solution

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

$$
\begin{aligned}
\mathsf{f}(\mathsf{end}) &= \mathsf{end} & \mathsf{f}(X) &= \overline{X} \\
\mathsf{f}(?T.S) &= !T.\mathsf{f}(S) & \mathsf{f}(\overline{X}) &= X \\
\mathsf{f}(!T.S) &= ?T.\mathsf{f}(S) & \mathsf{f}(\mu X.S) &= \mu X.(\mathsf{f}(S)\{\overline{X}/X\})
\end{aligned}
$$

Caveat

- The operation $\ldots\{\overline{X}/X\}$ is *not* standard substitution.
- It rather swaps X and $\overline{X}$.

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

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

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 upcoming paper at the PLACES 2020 workshop

Thank you!