Service Equivalence via Multiparty Session Type Isomorphisms

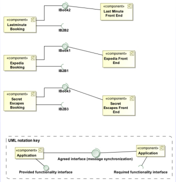
Assel Altayeva

December 19, 2019

ABCD meeting
Glasgow

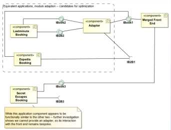
Setting

Interoperability of
Service Oriented Architecture



Matching:

Service specification

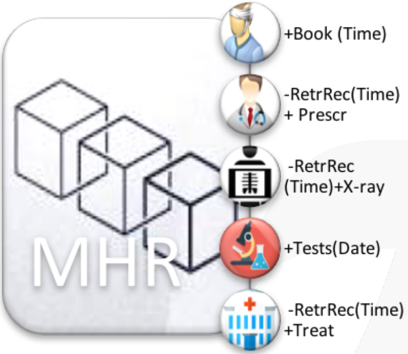


Bridging:

Constructive adaptors

Interoperability is a fundamental problem in software design, arising in various contexts (reuse, integration and legacy services)

Motivation: Medical Health Record



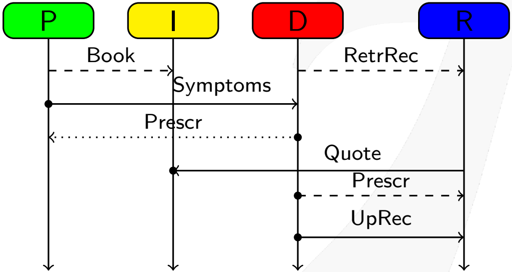
- Distributed Ledger Technology smart contract correctness does not have a formal verification framework.
- What is the optimal way to structure communication to ensure data provenance and safety?

Choreography of a medical health records system

- Industrial component based systems consist of a choreography of modules in which data items often possess a critical identity across their journey(Patient's Health Record Privacy and Accessibility).
- If provenance of data is formalised as traceability of items, then the expectations of provenance are formalised by a notion of component interfaces and component composition that can predicate over the journey of data items.
- We understand reuse and adaptability in terms of global choreographies of messages between components, considered as sessions across processes.

Distributed communication protocol- Health Record

- Four independent interfaces (Patient, Insurance, Doctor, Hospital Record)
- Structured protocol according to
  - rules of interactions (ex. Prescribe, Quote)
  - local contract conditions (Patient-Insurance)
  - accessibilty (ex. Secure record)
- Message-passing peer-to-peer communication
- No global control



Problem

- Global Choreography combines interface behaviours
- Question: How to verify correctness of the overall global choreography up to an equivalence (isomorphism)?
- Goal:
  - certified substitutability within a global choreography
  - provide interaction success (no orphan messages, deadlocks)
- Method: Multiparty Session Types (MPST)
  - Type theory for channel-based π-calculus
  - Global interaction choreographies between several participants
  - Local type enforcement to guarantee global progress (according to the specification).
  - Session refinement: enforcing other properties (security, state)

Type isomorphisms in Functional Programming

The notion of conversion, or adaptation without loss of information between types is commonly known as an isomorphism between the two types [1]

$$\begin{tikzcd} A \arrow[loop left, "id_A"] \arrow[rr, bend left, "f"] & & B \arrow[ll, bend left, "g"] \arrow[loop right, "id_B"] \end{tikzcd}$$

Two types are isomorphic ($A \cong B$) if there are mutually inverse procedures for transforming data between them.

[1] R. Di Cosmo. Isomorphisms of types: from λ-calculus to information retrieval and language design., 1995

Type isomorphisms practices: Functional Programming

- **Types as search keys**: using type isomorphism as a key tool for retrieving library components. (HOOGLE)
- **Inside Type Systems**: performing transformations of data types inside a programming language via isomorphism.(Mockingbird)
- **Building Coercions**:defining glue code in order to adapt to different contexts and language constructs(classes, objects and modules, dependent types in proof assistants)

Approaches to behavioral equivalences

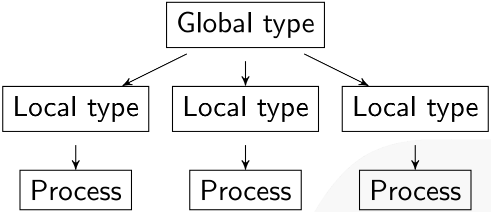
- **Bisimulation** two systems are able to mimic each others behaviour stepwise.[2]
- **Testing** two systems are considered to be equivalent if an external observer cannot distinguish between them. [3]
- **Trace** considers the computations of the systems taken in isolation, thus abstracting from the branching points of their behaviour.[4]

[2] D. Kouzapas, N. Yoshida:Globally Governed Session Semantics. LMCS, 2014

[3] G. Bernardi, M. Hennessy: Mutually Testing Processes. LMCS 2015

[4] R. Demangeon, N. Yoshida: On the Expressiveness of Multiparty Sessions. FSTTCS 2015

Multiparty session types[5]



- Structured communications from a global point of view, for example:

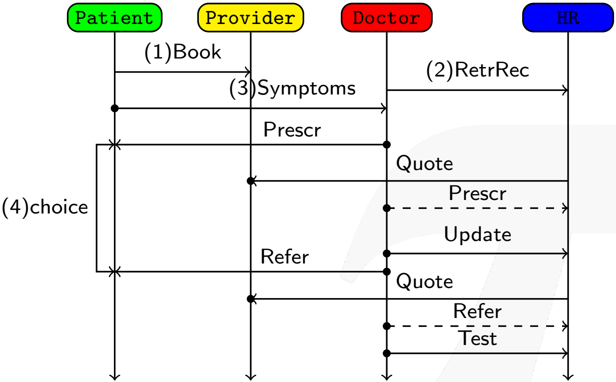
$$\mathsf{G} = A \to B : m_1; B \to C : m_2; A \to C : m_3.end$$

- Type-checking strategy of processes through projection of global types onto participants :

$$\mathsf{G} \restriction \mathsf{B} = A?m_1; C!m_2; end$$

[5] K. Honda, N.Yoshida, M. Carbone: Multiparty asynchronous session types.

GP visit protocol



Multiparty session type for NHS GP visit protocol

$G_{NHS} =$

(1) $\quad \mathsf{Patient} \to \mathsf{NHS} : \langle \mathsf{Book} \rangle;$

(2) $\quad \mathsf{Doctor} \to \mathsf{HR} : \langle \mathsf{RetrRec} \rangle;$

(3) $\quad \mathsf{Patient} \to \mathsf{Doctor} : \langle \mathsf{Symptoms} \rangle;$

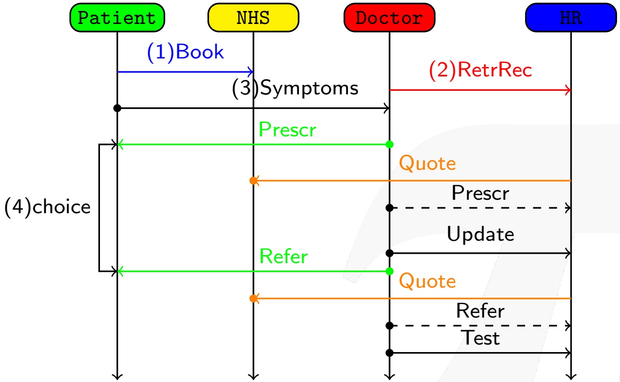
(4) $\quad \mathsf{Doctor} \to \mathsf{Patient} : \{\mathsf{Prescr}: \; \mathsf{HR} \to \mathsf{NHS} : \langle \mathsf{Quote} \rangle;$

$\quad \mathsf{Doctor} \to \mathsf{HR} : \{\mathsf{Prescr} : \mathsf{Doctor} \to \mathsf{HR} : \langle \mathsf{Update} \rangle; \mathsf{end}\},$

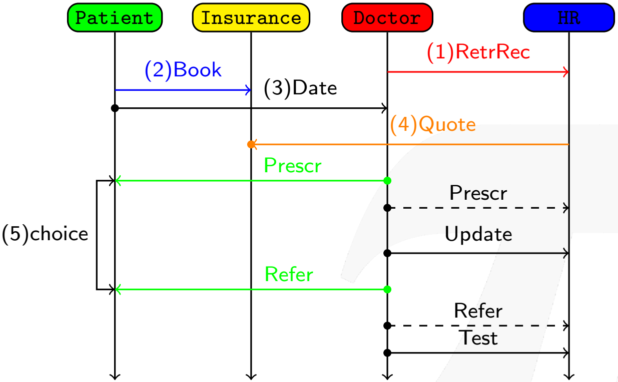
$\quad \mathsf{Refer} : \; \mathsf{HR} \to \mathsf{NHS} : \langle \mathsf{Quote} \rangle;$

$\quad \mathsf{Doctor} \to \mathsf{HR} : \{\mathsf{Refer} : \mathsf{Doctor} \to \mathsf{HR} : \langle \mathsf{Test} \rangle; \mathsf{end}\} \; \}.$

NHS GP visit protocol



Private GP visit protocol



Candidate for equivalent multiparty session type

$G_{Private} =$

$$
\begin{array}{ll}
(1) & \mathsf{Doctor} \to \mathsf{HR} : \langle \mathsf{RetrRec} \rangle; \\
(2) & \mathsf{Patient} \to \mathsf{Insurance} : \langle \mathsf{Book} \rangle; \\
(3) & \mathsf{Patient} \to \mathsf{Doctor} : \langle \mathsf{date} \rangle; \\
(4) & \mathsf{HR} \to \mathsf{Insurance} : \langle \mathsf{quote} \rangle; \\
(5) & \mathsf{Doctor} \to \mathsf{Patient} : \{\mathsf{Prescr} : \\
& \quad \mathsf{Doctor} \to \mathsf{HR} : \{\mathsf{Prescr} : \mathsf{Doctor} \to \mathsf{HR} : \langle \mathsf{Update} \rangle; \mathsf{end}\}, \\
& \qquad \mathsf{Refer} : \\
& \quad \mathsf{Doctor} \to \mathsf{HR} : \{\mathsf{Refer} : \mathsf{Doctor} \to \mathsf{HR} : \langle \mathsf{Test} \rangle; \mathsf{end}\}\}.
\end{array}
$$

Are these two global types equivalent?

$$G_{NHS} \cong_{?} G_{Private}$$

Type theoretic behavioural equivalence example

An interface **type** for an online banking system's login

```
string login(string username, int pin)
```

can be considered **isomorphic** to

```
string login2(int pin, string username)
```

because we can **convert** or **adapt** code that satisfies the first type to match the second, and vice versa:

```
string login2(int pin, string username) { return
login(username, pin); }
```

Isomorphism and invertibility

The study of the type isomorphisms in λ-calculus is based on the notion of λ-term invertibility. Dezani fully characterized invertible λ-terms in [6] as the finite hereditary permutators, λ-terms of the form

$$\lambda x y_1..y_n.x(P_1 y_{\pi(1)})...(P_n y_{\pi(n)})(n \geq 0)$$

where π is a permutation of $1,..,n$, and $P_1,...,P_n$ are FHPs.

(Ax 1)	$A \times B \cong B \times A$
(Ax 2)	$A \times (B \times C) \cong (A \times B) \times C$
(Ax 3)	$(A \times B) \to C \cong A \to (B \to C)$
(Ax 4)	$A \to (B \times C) \cong (A \to B) \times (A \to C)$

Table: Some type isomorphisms axioms for the First order λ -calculus

[6] M. Dezani-Ciancaglini: Characterization of Normal Forms Possessing Inverse in the lambda-beta-eta-Calculus. TCS 2(3): 323-337 (1976)

How to axiomatise multiparty session type isomorphism in the context of adaptation?

Approach

- Trace-based (denotational) models of session types to compare expressiveness of sessions.
- Λ-term combinators over syntactic structure of the global type.
- Logical specifications to impose restrictions.

Multiparty session types syntax

Participants $\mathrm{p}, \mathrm{q}..$
Types of exchanged messages $U \in \{\mathsf{Bool}, \mathsf{Int}\}$
Labels $l_1, ..., l_n$
Prefix

$$g ::= \mathrm{p} \to \mathrm{q} : \langle U \rangle$$

$\mathsf{inp}(g) := \mathrm{q}$, $\mathsf{out}(g) := \mathrm{p}$
$\mathsf{pid}(g) = \{\mathrm{p}, \mathrm{q}\}$;
Branch Prefix

$$g_i ::= \mathrm{p} \to \mathrm{q} : l_i, \forall i \in I$$

$\mathsf{inp}(g_i) := \mathrm{q}$, $\mathsf{out}(g_i) := \mathrm{p}$ $\forall i \in I$
$\mathsf{pid}\ (\mathrm{g}_i) = \{\mathrm{p}, \mathrm{q}\}$.

Multiparty session type syntax

U	$::=$	Bool $\mid$ Int	Value types
	Global types		
Gtype	$::=$	$\mathsf{g}; G$	Prefix
		$g_1; G_1 \times \ldots \times g_k; G_{k, k \in I}$	Branching
		$\mu \mathbf{t}.G \;\; \mathbf{t} \;\;$ end	Recursion/end
	Local session types		
T	$::=$	$\mathsf{inp(g)!}\langle U \rangle; T$	Send
		$\mathsf{out(g)?}\langle U \rangle; T$	Receive
		$\mathsf{inp(g)} \oplus \{l_i; T_i\}$	Branching
		$\mathsf{out(g)}\ \&\ \{\mathsf{l}_i; T_i\}$	Selection
		$\mu \mathbf{t}.T \;\; \mathbf{t} \;\; \Vert$ end	Recursion/end

Operational semantics for global types

$$g; G \xrightarrow{g} G \qquad \text{[Inter]}$$

$$g_1; G_1 \times \ldots \times g_i; G_{i, i \in I} \xrightarrow{g_k} G_k \qquad \text{[SelBra]}$$

$$\frac{G \xrightarrow{g'} G' \qquad empty_S(g, g')}{g; G \xrightarrow{g'} g; G'} \qquad \text{[IPerm]}$$

$$\frac{\forall i \in I, G_i \xrightarrow{g'} G'_i \qquad empty_S(g', g_i)}{g_1; G_1 \times \ldots \times g_i; G_{i, i \in I} \xrightarrow{g'} g_1; G'_1 \oplus \ldots \oplus g_n; G'_{i, i \in I}} \qquad \text{[SBPerm]}$$

$$\frac{G[\mu t.G/t] \xrightarrow{g} G'}{\mu t.G \xrightarrow{g} G'} \qquad \text{[Rec]}$$

Semantic view of the MPST isomorphism: globally

Trace of a global type

Given global type G, we call the trace of a global type a sequence of possible communication events during protocol execution:

$$Tr(G) = \{g_1; g_2..; g_n | G \xrightarrow{g_1} .. \xrightarrow{g_n} G', g_{i\in I} : \text{Prefix}\}$$

Synchronous semantics

$$\text{empty}_S(g, g') = \begin{cases} \text{True}, & \text{pid(g)} \cap \text{pid(g}') = \emptyset, \\ \text{False}, & \text{else.} \end{cases} \tag{1}$$

Ex.

$$\overbrace{p \to q : l_1}^{g_1} ; \underbrace{r \to s : l_2}_{g_2} . \overbrace{w \to z : l_3}^{g_3} .end$$

$$(1) \xrightarrow{g_1} g_2 ; g_3 ; end \xrightarrow{g_2} g_3 ; end \xrightarrow{g_3} end$$

$$(2) \xrightarrow{g_2} g_1 ; g_3 ; end \xrightarrow{g_3} g_1 ; end \xrightarrow{g_1} end$$

$$(3) \xrightarrow{g_3} g_1 ; g_2 ; end \xrightarrow{g_1} g_2 ; end \xrightarrow{g_2} end$$

Semantic view of the MPST isomoprhism: locally

[LIn]	$\mathrm{out}(g)?\langle U\rangle; T \xrightarrow{\mathrm{out}(g)?\langle U\rangle} T$
[LOut]	$\mathrm{inp}(g)!\langle U\rangle; T \xrightarrow{\mathrm{inp}(g)!\langle U\rangle} T$
[LBra]	$\mathrm{out}(g)\&\{l_i : T_i\} \xrightarrow{\mathrm{out}(g)?l_j} T_j \quad (j \in I)$
[LSel]	$\mathrm{inp}(g) \oplus \{l_i : T_i\} \xrightarrow{\mathrm{inp}(g)!l_j} T_j \quad (j \in I)$
[LRec]	$T[\mu \mathbf{t}.T/\mathbf{t}] \xrightarrow{\ell} T' \implies \mu \mathbf{t}.T \xrightarrow{\ell} T', \quad \ell \in \mathfrak{L}$

Table: Operational Semantics of Local Types

where

$\mathfrak{L} = \{\mathrm{inp}(g)!m, \mathrm{out}(g)?m \quad | \quad m \in \{\langle U\rangle, l\}, g : \mathrm{Prefix}, U : \mathrm{VType}, l : \mathrm{Label}\}$

Semantic view of the MPST isomoprhism: locally

Configuration traces

A configuration trace σ is a mapping from participants to a sequence of labels of local types, i.e. $\sigma(\mathrm{r}) = \ell_1 ... \ell_n$ where $\ell_i \in \mathfrak{L}$. A participant r is in the domain of σ if $\sigma(\mathrm{r}) \neq \varepsilon$ where ε stands for an empty sequence.

Denotation of a MPST[7]

Denotation of a global type and terminated traces

Let us define $\delta(G) = (T_{\mathrm{p}})_{\mathrm{p} \in \mathscr{P}}$ where $\mathscr{P}$ is a set of participants in G. We define the denotation of global type G under synchronous semantic, denoted $\mathbf{D}(G)$, as the set of all terminated traces from $\delta(G)$ where a terminated trace from $\delta(G)$ means $\delta(G) \rightsquigarrow^{\sigma}_{\text{synch}} \Delta$ where $\Delta \not\rightarrow$.

[7] R. Demangeon, N. Yoshida. On the expressiveness of multiparty sessions. FSSTCS(2015)

Equational relation for synchronous global types through trace semantics

Theorem (Equivalence between Synchronous Global Types and Configuration Traces)

Let G be a global type with participants $\mathscr{P}$ and let $\Delta = (G \upharpoonright \mathrm{p})_{\mathrm{p} \in \mathscr{P}}$ be the local type configuration projected from G. Then $Tr(G) \equiv \mathscr{T}_S(\Delta)$ where $\Delta = (T_{\mathrm{p}})_{\mathrm{p} \in \mathscr{P}}$.

Isomorphism and trace set equivalence for synchronous semantics.

Lemma

If $G_1 \rightleftarrows^{SBD} G_2$ then $\mathbf{D}(G_1) \equiv \mathbf{D}(G_2)$

Definable isomorphism $\rightleftarrows^{SBD}$:

- Swapping
- Branching
- Distributivity

Syntactic view of the MPST isomorphism

Global type definable isomorphism

Two global types G and G' are isomorphic $G \rightleftarrows G'$ iff there exist combinators $M(G) = G'$ and $N(G') = G$, such that $\mathbf{D}(G) \equiv \mathbf{D}(G')$, where M, N are compositions of combinators.

In order to build isomorphism combinators we require two syntax classes of variables:

λ-terms over MPST Syntax

$$
\begin{array}{llll}
(\textsc{Variables}) & \mathsf{v} & := & \mathsf{v}_g : \mathsf{Prefix} \mid \mathsf{v}_G : \mathsf{Gtype} \\
(\Lambda\textsc{-terms}) & M & := & \mathsf{v} \mid \lambda \mathsf{v}.M \mid \text{if } e \text{ then } M \text{ else } M \mid \\
 & & & \text{let } \mathsf{v} = M \text{ in } M \mid MM
\end{array}
$$

$(\textsc{Boolean})$

$$e := \text{true} \mid \text{false} \mid \text{not}(e) \mid e_1 \text{ and } e_2 \mid e_1 \text{ or } e_2$$

Combinators

Prefix commutativity

$$G = g_1; ..; g_{i-1}; g_i; ..g_n; \overline{G} \underset{Swap^r_{g_i}}{\overset{Swap^l_{g_i}}{\rightleftarrows}} g_1; ..; g_{i-2}; g_i; g_{i-1}..g_n; \overline{G}$$

Where

$$\mathrm{Swap}^{\mathrm{l}}_{\mathrm{g_i}} \triangleq \lambda\, G : \mathrm{Gtype.\ let} \quad g_i = F_i(G) \text{ and } G' = \mathrm{Tail}_i(G) \quad \text{in}$$
$$\text{if} \quad \mathrm{pid}(g_i, g_{i-1}) = \emptyset \quad \text{then} \quad g_1; ..; g_{i-2}; g_i; g_{i-1}; G' \quad \text{else} \quad G$$

$$\mathrm{Swap}^{\mathrm{r}}_{\mathrm{g_i}} \triangleq \lambda\, G : \mathrm{Gtype.\ let} \quad g_i = F_i(G) \text{ and } G' = \mathrm{Tail}_{i+1}(G) \quad \text{in}$$
$$\text{if} \quad \mathrm{pid}(g_i, g_{i+1}) \quad \text{then} \quad g_1; ..; g_{i-1}; g_{i+1}; g_i; G' \quad \text{else} \quad G$$

Branching

Branching

$$g_1; g; G_1 \times \ldots \times g_i; g; G_i \rightleftarrows^{Contr}_{Exp} g; (g_1; G_1 \times \ldots \times g_i; G_{i, i \in I})$$

$$\begin{aligned}\mathrm{Contr(G)} \triangleq \lambda G_1 \ldots \lambda G_k. \quad \text{if} \quad & G = g_1; g; G_1 \times \ldots \times g_k; g; G_k \quad \text{and} \\ & \mathrm{empty}_\star(g, g_i), 1 \leq i \leq k \\ \text{then} \quad & g; (g_1; G_1 \oplus \ldots \oplus g_k; G_k) \quad \text{else} \quad G\end{aligned}$$

$$\begin{aligned}\mathrm{Exp(G)} \triangleq \lambda G_1 \ldots \lambda G_k. \quad \text{if} \quad & G = g; (g_1; G_1 \oplus \ldots \oplus g_k; G_k) \quad \text{and} \\ & \mathrm{empty}_\star(g, g_i), 1 \leq i \leq k \\ \text{then} \quad & g_1; g; G_1 \times \ldots \times g_k; g; G_k \quad \text{else} \quad G\end{aligned}$$

Distributivity

Branching within Branches

$$g_1;(\overline{g}_{n+1};G_1\times\ldots\times\overline{g}_{n+k};G_k)\oplus\ldots\oplus g_n;(\overline{g}_{n+1};G_1\times\ldots\times\overline{g}_{n+k};G_k)\ \underset{SwapBr_r}{\overset{SwapBr_l}{\rightleftarrows}}$$

$$\overline{g}_{n+1};(g_1;G_1\oplus\ldots\oplus g_n;G_1)\times\ldots\times\overline{g}_{n+k};(g_1;G_k\oplus\ldots\oplus g_n;G_k),\ \ k\in I,n\in I\quad \text{else}\quad G.$$

$$\mathrm{SwapBr_l(G)}\triangleq\lambda g_1\ldots\lambda g_n\lambda\overline{g}_{n+1}\ldots\lambda\overline{g}_{n+k}\lambda G_1\ldots\lambda G_k.$$

$$\text{if}\quad G=\bigoplus_{i\in I}G_i'\quad\text{and}\quad G_i'=g_i;(\overline{g}_{n+1};G_1\times\ldots\times\overline{g}_{n+k};G_k),$$

$$\text{then}\quad \mathrm{Exp}(G_i')n\in I,k\in I\quad\text{else}\quad G.$$

$$\mathrm{SwapBr_r(G)}\triangleq\lambda g_1\ldots\lambda g_n\lambda\overline{g}_{n+1}\ldots\lambda\overline{g}_{n+k}\lambda G_1\ldots\lambda G_k.$$

$$\text{if}\quad G=\bigoplus_{i\in 1..k}G_i'\quad\text{and}\quad G_i=\overline{g}_{n+i};(g_1;G_i\oplus\ldots\oplus g_n;G_i)$$

$$\text{then}\quad \mathrm{Contr}(G_i')n\in I,k\in I\quad\text{else}\quad G.$$

Soundness

Theorem

Let G be a global type with participants $\mathscr{P}$. If $G_1 \rightleftarrows G_2$, then $\mathscr{T}_S(\Delta_1) = \mathscr{T}_S(\Delta_2)$ where $\Delta_i = (T_{i\mathrm{p}})_{\mathrm{p}\in\mathscr{P}}$ with $i \in \{1,2\}$ and $T_{i\mathrm{p}} = G_i \upharpoonright \mathrm{p}$. Hence if $G_1 \rightleftarrows G_2$, then $\mathbf{D}(G_1) = \mathbf{D}(G_2)$.

Related work - binary session type isomorphisms

- Types are formulas of intuitionistic logic [8] - isomorphism of types is isomorphism in linear logic:

$$A \otimes B \cong B \otimes A$$

$$A \multimap (B \multimap C) \cong (A \otimes B) \multimap C$$

- Session type isomorphism for two-channel adjacent processes [9]

$$!t.!s.T \cong !s.!t.T$$

$$?t.(T + S) \cong ?t.T + ?t.S$$

[8] J. A. Perez, L. Caires, F. Pfenning, B. Toninho: Linear logical relations and observational equivalences for session-based concurrency. Inf. Comput. 239: 254-302 (2014)

[9] M. Dezani-Ciancaglini, L. Padovani, J. Pantovic: Session type isomorphism. PLACES(2014)

Next steps

- Investigation of global trace semantics for asynchronous MPST.
- Completeness by enriching isomorphism axiom system.
- Practical applications of session type isomorphism to asynchronous/synchronous multi-party processes.
- Scribble Protocol Description language library search tool.