Service Equivalence
via Multiparty Session Type Isomorphisms

Assel Altayeva   Nobuko Yoshida

Imperial College London, UK

This paper addresses a problem found within the construction of Service Oriented Architecture: the adaptation of service protocols with respect to functional redundancy and heterogeneity of global communication patterns. We utilise the theory of Multiparty Session Types (MPST). Our approach is based upon the notion of a multiparty session type isomorphism, utilising a novel constructive realisation of service adapter code to establishing equivalence. We achieve this by employing trace semantics over a collection of local types and introducing meta abstractions over the syntax of global types. We develop a corresponding equational theory for MPST isomorphisms. The main motivation for this line of work is to define a type isomorphism that affords the assessment of whether two components/services are substitutables, modulo adaptation code given software components formalised as session types.

1   Introduction

Multiparty session types (MPST) [10] formalise multi component distributed architectures whose semantics is necessarily given as message passing choreographies. The desired interactions are specified into a session or a *global* type between participants through a series of simple syntax including interaction between two participants (composition of one send and corresponding receive), choice and recursion. Global types are then projected onto *local* types, describing communication from each participant's point of view. The theory of session types guarantees that local conformance of all participants results in of an architecture that globally conforms to the initially specified global types.

We follow the approach developed in [8]. In that work, the notion of session type isomorphisms was initially explored. The main motivation for that line of work is to define the type isomorphism that would allow assessment of whether two components/services are substitutable modulo adaptation code, given component specification is considered to be a session type. This approach to isomorphism practically consists of a library of constructive combinators for witnessing this kind of equivalence. We build upon this with the intention of defining multiparty session type isomorphism combinators and study their correctness. Comparison of two MPST is double layered: there is global syntax that is grounded in local communication semantics.

The common framework involves both *configurations* (collections of local types) and traces of events performed in the course of the global protocol execution as in [6]. Hence syntactic change to the global protocol description might not affect local types and affords candidates for equivalent MPST. To construct combinators for MPST isomorphisms, we employ meta abstractions over global syntax, ensuring the isomorphism preserves MPST well-formedness (projectability to local types, guaranteeing there are no orphan messages, deadlocks and that each participant has unambiguous instruction for the behaviour within the protocol). We find concurrent interactions that are independent and that do not change the outcome if permuted.

$G =$
  $(1)\ P \rightarrow I : \langle PId, DId \rangle;$
  $(2)\ D \rightarrow R : \langle RetrRec \rangle;$
  $(3)\ P \rightarrow D : \langle IId, Symptoms \rangle;$
  $(4)\ D \rightarrow P : \{ Prescr : R \rightarrow I : \langle Quote \rangle;$
        $D \rightarrow R : \{ Prescr : D \rightarrow R : \langle UpRec \rangle; end \},$
             $Ref : R \rightarrow I : \langle Quote \rangle;$
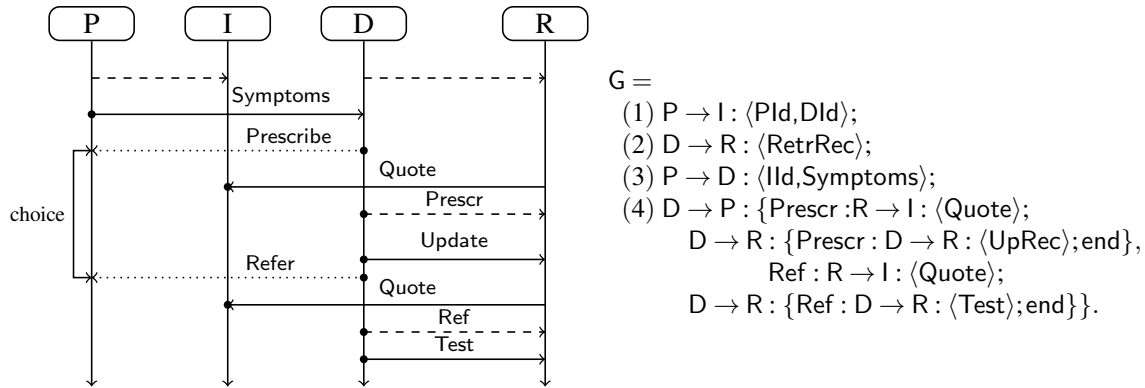 $D \rightarrow R : \{ Ref : D \rightarrow R : \langle Test \rangle; end \} \}.$

Figure 1: eHealth GP Visit Protocol

eHealth Record Example   Consider the following example, a basic eHealth record logging system. Communication is between four participants: Patient (P), Doctor (D), Insurance company (I) and Hospital Record (R). The diagram and global type protocol for it are depicted in Fig.1. The insurance company is required to give approval according to their contract conditions at each step of the treatment and at the same time hospital records have to be updated and available for further hospital and specialist systems. There are independent communications happening between Doctor and Patient, while the Insurance company makes enquiries with Hospital Records.

Let us discuss the protocol in the Figure 1. First, $P \rightarrow I : \langle PId, DId \rangle$ global type describes Patient booking appointment with the Insurance by sending his/her identification of the type PId and Doctor's details of DId value type. In the second line of the global type G we have $D \rightarrow R : \langle RetrRec \rangle$, Doctor opens Patients records with the Hospital Record system, sending retrieval protocol of type RetrRec. Next the Patient lists his "Symptoms" to the Doctor, who either prescribes a treatment or refers the Patient for further tests. At line (4), the Doctor sends his choice to the Patient and updates Patient's Hospital Record, initiating Referral or Prescription protocol with Insurance company and the Hospital.

Now we observe that lines (1) and (2) follow communication between pairwise different participants, which means there is no synchronisation dependence between these two communications and they could be swapped. Another candidate for protocol transformation is the exchange between the Insurance and the Hospital Record (line (4)), triggered by the Doctor-to-Patient communication where Insurance and the Hospital Record are not aware of the order and content of the branching choices.

In this work we restrict our consideration to a synchronous setting, when participants cannot start a new interaction without completing current one by waiting for the message to be received. Hence global protocol transformation will stem from the communications, which are order dependent or independent(between pairwise different participants). In the next section, we will define the formal setting for protocol transformation combinators.

2   Design of Multiparty Session Type Isomorphism

Global types define overall schemes of labelled communications between session participants. We will deconstruct the general global type syntax introduced in [3] to include Prefix and Branch subterms of the global type denoted Gtype. We assume base set of *participants*, ranged over by $p, q, r, s..$; *exchange values*, ranging over boolean or natural numbers, which could also be assigned to *labels* $l_1, l_2, ..$ and *recursion variables*, ranged over $\mathbf{t}, \mathbf{t}'...$

$$\frac{}{\vdash \mathsf{p} : \mathsf{Part}} \text{ (Participant)} \qquad \frac{\vdash U : \mathsf{Bool}}{\vdash U : \mathsf{Vtype}} \qquad \frac{\vdash U : \mathsf{Nat}}{\vdash U : \mathsf{Vtype}} \text{ (Exchange Values)} \qquad \frac{\vdash U : \mathsf{Vtype}}{\vdash U : \mathsf{Label}} \text{ (Labels)}$$

$$\frac{\vdash \mathsf{p},\mathsf{p}' : \mathsf{Part} \quad \vdash U : \mathsf{VType}}{\vdash \mathsf{p} \to \mathsf{p}' : \langle U \rangle : \mathsf{Prefix}} \text{ (Global Prefix)}$$

$$\frac{}{\vdash \mathsf{end} : \mathsf{Gtype}} \qquad \frac{}{\vdash \mathbf{t} : \mathsf{Gtype}} \qquad \frac{\vdash g : \mathsf{Prefix} \quad \vdash G : \mathsf{Gtype}}{\vdash g; G : \mathsf{Gtype}} \text{ (Global type)}$$

$$\frac{\vdash \mathsf{p},\mathsf{p}' : \mathsf{Part} \quad \vdash l_i : \mathsf{Label} \quad \vdash G_i : \mathsf{Gtype} \quad \vdash g_i := \mathsf{p} \to \mathsf{p}' : l_i : \mathsf{Prefix} \quad i \in I}{\vdash g_1; G_1 \times \ldots \times g_i; G_{i, i \in I} : \mathsf{Gtype}} \text{ (Branching)}$$

$$\frac{\vdash \mathbf{t} : \mathsf{Gtype} \quad \vdash G : \mathsf{Gtype}}{\vdash \mu \mathbf{t}.G : \mathsf{Gtype}} \text{ (Recursion)}$$

Table 1: Formation Rules for Global Types

Type formation rules for the multiparty global type abstraction are outlined in the Table 1.

Global Types We define the syntax for MPST in the Definition 1, introducing prefix terms for pairwise communications and predicates inp and out over it that distinguish inputting and outputting participants, which will be useful for developing global and local type trace based understanding of equivalence between two MPST.

Definition 1 (*Multiparty Session Types*). *Given participants* $\mathsf{p}, \mathsf{q}..$, *types of exchanged messages* $U \in \{\mathsf{Bool}, \mathsf{Int}\}$ *and labels* $l_1, ..., l_n$, *the grammar of global types* $(G, G'..)$ *is defined as:*

$$G \quad ::= \quad g; G \quad | \quad g_1; G_1 \times \ldots \times g_k; G_{k, k \in I} \quad | \quad \mathbf{t} \quad | \quad \mu \mathbf{t}.G \quad | \quad \mathsf{end}$$
$$g \quad ::= \quad \mathsf{p} \to \mathsf{q} : \langle U \rangle \qquad g_i \quad ::= \quad \mathsf{p} \to \mathsf{q} : l_i, \forall i \in I$$
$$\mathsf{inp}(g) := \mathsf{q}, \mathsf{out}(g) := \mathsf{p} \text{ with } \mathsf{pid}(g) = \{\mathsf{p}, \mathsf{q}\}; \text{ and } \mathsf{inp}(g_i) := \mathsf{q}, \mathsf{out}(g_i) := \mathsf{p} \text{ with } \forall i \in I.\mathsf{pid}(g_i) = \{\mathsf{p}, \mathsf{q}\}.$$

The corresponding local session types syntax is as follows:

$$T \quad ::= \quad \mathsf{inp}(g)!\langle U \rangle; T \quad | \quad \mathsf{out}(g)?\langle U \rangle; T \quad | \quad \mathsf{inp}(g_i) \oplus \{l_i : T_i\} \quad | \quad \mathsf{out}(g_i) \& \{l_i : T_i\} \quad | \quad \mathbf{t} \quad | \quad \mu \mathbf{t}.T \quad | \quad \mathsf{end}$$

We will redefine the standard notion of the projection from global to local types. Mergeability $\bowtie$ is the smallest equivalence over local types closed under all contexts and the mergeability rule. If $T_1 \bowtie T_2$ holds then branch merging is well-defined with a partial commutative operator $\sqcup$. See [7] for the full definition.

Definition 2 (*Global Type Projection*) *Projection* $(G \upharpoonright \mathsf{q})$ *of a global type* G *onto a participant* q *is defined by induction on* G. *Let* $g = \mathsf{p} \to \mathsf{p}' : \langle U \rangle$ *and* $g_i = \mathsf{p} \to \mathsf{p}' : l_{i, \forall i \in I}$:

$$(g; G') \upharpoonright \mathsf{q} = \begin{cases} \mathsf{inp}(g)!\langle U \rangle; (G' \upharpoonright \mathsf{q}) & \text{if} \quad \mathsf{q} = \mathsf{out}(g) \\ \mathsf{out}(g)?\langle U \rangle; (G' \upharpoonright \mathsf{q}) & \text{if} \quad \mathsf{q} = \mathsf{inp}(g) \\ G' \upharpoonright \mathsf{q} & \text{otherwise} \end{cases}$$

$$(g_1; G_1 \times \ldots \times g_n; G_n)_{n \in I} \upharpoonright \mathsf{q} = \begin{cases} \mathsf{inp}(g) \oplus \{l_i : (G_i \upharpoonright \mathsf{q})\}_{i \in I} & \text{if } \mathsf{q} = \mathsf{out}(g_i)_{i \in I} \\ \mathsf{out}(g) \& \{l_i : (G_i \upharpoonright \mathsf{q})\}_{i \in I} & \text{if } \mathsf{q} = \mathsf{inp}(g_i)_{i \in I} \\ \sqcup_{i \in I} G_i \upharpoonright \mathsf{q} & \text{if } \mathsf{q} \neq \mathsf{pid}(g_i)_{i \in I} \quad \text{and} \\ & \forall i, j \in I.G_i \upharpoonright \mathsf{q} \bowtie G_j \upharpoonright \mathsf{q} \end{cases}$$

$$(\mu t.G) \restriction \mathsf{q} = \begin{cases} \mu t.(G \restriction \mathsf{q}) & \text{if } G \restriction \mathsf{q} \neq t, \\ \text{end} & \text{otherwise}. \end{cases} \qquad t \restriction \mathsf{q} = t \qquad \text{end} \restriction \mathsf{q} = \text{end}.$$

Within our example, projection onto the Insurance participant reflects his/her ignorance of the choice sent by the Doctor to the Patient, therefore Insurance is activated by the Patient sending it booking details, waits for the Quote from the Health Records and then ends: $G \restriction \mathsf{I} = \mathsf{P}?\langle \mathsf{PId}, \mathsf{DId}\rangle; \mathsf{R}?\langle \mathsf{Quote}\rangle; \text{end}$.

The operational semantics for local types is defined in Table 2 with local labels set ranged over by $\ell, \ell', \ldots$:

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

where $\mathsf{inp}(g)!m$ is a send action (participant $\mathsf{out}(g)$ is sending m, which could be a value or a label, to participant $\mathsf{inp}(g)$) and $\mathsf{out}(g)?m$ is a dual receive action.

[LIn] $\mathsf{out}(g)?\langle U\rangle; T \xrightarrow{\mathsf{out}(g)?\langle U\rangle} T$ [LOut] $\mathsf{inp}(g)!\langle U\rangle; T \xrightarrow{\mathsf{inp}(g)!\langle U\rangle} T$

[LBra] $\mathsf{out}(g)\&\{l_i : T_i\} \xrightarrow{\mathsf{out}(g)?l_j} T_j \quad (j \in I)$ [LSel] $\mathsf{inp}(g) \oplus \{l_i : T_i\} \xrightarrow{\mathsf{inp}(g)!l_j} T_j \quad (j \in I)$

[LRec] $T[\mu t.T/t] \xrightarrow{\ell} T' \implies \mu t.T \xrightarrow{\ell} T', \quad \ell \in \mathfrak{L}$

Table 2: Operational Semantics of Local Types

[LIn] is for a single receive action and its dual [LOut] for a send action. Similarly, [LSel] is the rule for sending a label and its dual [LBra] is for receiving a label. Rule [LRec] is the standard rule for recursions.

Reduction rules for the global protocol are summarised in Table 3. This work focuses upon synchronous semantics for the global type communication, following methodology introduced in [11]. [Inter] shows reduction of the global type when communication within prefix g occurs, similarly [SelBra] rule shows selecting of one of the branches g_k and executing it, which results in reduction of the global type to the continuation of the selected branch G_k. Rules [IPerm] and [SBPerm] show how to execute message passing between participants that are not part of the the prefix communication. The last rule [Rec] is the standard rule for recursions.

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

[IPerm] $\dfrac{G \xrightarrow{g'} G' \quad \mathsf{pid}(g) \cap \mathsf{pid}(g') = \emptyset}{g; G \xrightarrow{g'} g; G'}$ [Rec] $G[\mu t.G/t] \xrightarrow{g} G' \implies \mu t.G \xrightarrow{g} G'$

[SBPerm] $\dfrac{\forall i \in I, G_i \xrightarrow{g'} G'_i \quad \mathsf{pid}(g') \cap \mathsf{pid}(g_i) = \emptyset}{g_1; G_1 \times \ldots \times g_i; G_{i,i\in I} \xrightarrow{g'} g_1; G'_1 \times \ldots \times g_i; G'_{i,i\in I}}$

Table 3: Operational Semantics of Global Types

Definition 3 *(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:*

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

λ-Terms of MPST   In order to build isomorphism combinators we require two syntactic classes of variables: one called Prefix for *term* variables, and another one called Gtype for global *type* variables. We define typed λ-terms on the variables of the Prefix or Gtype types:

(VARIABLES)	v	$:=$	$\mathsf{v}_g : \mathsf{Prefix}$	$\mathsf{v}_G : \mathsf{Gtype}$

(Λ-TERMS) $M \quad := \quad \mathsf{v} \quad | \quad \lambda \mathsf{v}.M \quad | \quad$ if e then M else $M \quad | \quad$ let $\mathsf{v} = M$ in $M \quad | \quad MM$

(BOOLEAN EXPRESSIONS) $e \quad := \quad$ true $\quad | \quad$ false $\quad | \quad$ not$(e) \quad | \quad e_1 \quad$ and $\quad e_2 \quad | \quad e_1 \quad$ or $\quad e_2$

We work with the usual λ-calculus typing judgment rules for well-formation. Isomorphisms are combinators (functions) with respect to abstraction over global session type expressions, i.e. Prefix for term variables and Gtype for global type variables.

We describe isomorphism in terms of invertible transformations over global types syntax as in the following definition:

Definition 4 *(Global Type Isomorphism and Invertible Combinators) Two global types G and G' are isomorphic $G \rightleftarrows G'$ iff there exist functions $M : G \rightarrow G'$ and $N : G' \rightarrow G$, such that $M \circ N = \lambda x : G.x$ and $N \circ M = \lambda x : G'.x$. Terms M,N are called invertible combinators.*

Let us assume $g_j = F_j(G), j \in I$ where combinator F_j produces j-th prefix and $\mathsf{Tail}_i(G) = G'$ with $j < i, i, j \in I$ and $\mathsf{Tail}_j(G) = g_{j+1}; g_{j+2}; ...; g_i; G'$, we can write a swapping combinator:

$$G = g_1; ..; g_{i-1}; g_i; ..g_n; \overline{G} \overset{\mathsf{Swap}^l_{g_i}}{\underset{\mathsf{Swap}^r_{g_i}}{\rightleftarrows}} g_1; ..; g_{i-2}; g_i; g_{i-1}..g_n; \overline{G} \qquad \textbf{(Prefix commutativity)}$$

where

$$\mathsf{Swap}^l_{g_i}(G) \triangleq \lambda G. \text{ let } \quad g_i = F_i(G) \text{ and } G' = \mathsf{Tail}_i(G) \quad \text{in} \tag{1}$$
$$\text{if } \mathsf{pid}(g_i) \cap \mathsf{pid}(g_{i-1}) = \emptyset \quad \text{then} \quad g_1; ..; g_{i-2}; g_i; g_{i-1}; G' \quad \text{else} \quad G.$$

and reverting combinator $\mathsf{Swap}^r_{g_i}$ will accordingly have a form:

$$\mathsf{Swap}^r_{g_i}(G) \triangleq \lambda G. \text{ let } \quad g_i = F_i(G) \text{ and } G' = \mathsf{Tail}_{i+1}(G) \quad \text{in} \tag{2}$$
$$\text{if } \mathsf{pid}(g_i) \cap \mathsf{pid}(g_{i+1}) = \emptyset \quad \text{then} \quad g_1; ..; g_{i-1}; g_{i+1}; g_i; G' \quad \text{else} \quad G.$$

Returning to our example protocol G from the Fig 1, if we apply this combinator to swap first two lines of independent communication we arrive at an isomorphic protocol G_{sw}:

$$\mathsf{G} \overset{\mathsf{Swap}^l}{\underset{\mathsf{Swap}^r}{\rightleftarrows}} \mathsf{D} \rightarrow \mathsf{R} : \langle \mathsf{RetrRec} \rangle; \mathsf{P} \rightarrow \mathsf{I} : \langle \mathsf{PId}, \mathsf{DId} \rangle; \mathsf{Tail}(\mathsf{G}) = \mathsf{G}_{sw}$$

.

$\mathsf{G}_{sw}^{br} =$
(1) $\mathsf{D} \to \mathsf{R} : \langle \mathsf{RetrRec} \rangle;$
(2) $\mathsf{P} \to \mathsf{I} : \langle \mathsf{PId}, \mathsf{DId} \rangle;$
(3) $\mathsf{P} \to \mathsf{D} : \langle \mathsf{IId}, \mathsf{Symptoms} \rangle;$
(4) $\mathsf{R} \to \mathsf{I} : \langle \mathsf{Quote} \rangle;$
(5) $\mathsf{D} \to \mathsf{P} : \{\mathsf{Prescr} : \mathsf{D} \to \mathsf{R} :$
        $\{\mathsf{Prescr} : \mathsf{D} \to \mathsf{R} : \langle \mathsf{UpRec} \rangle; \mathsf{end}\},$
                $\mathsf{Ref} : \mathsf{D} \to \mathsf{R} :$
        $\{\mathsf{Ref} : \mathsf{D} \to \mathsf{R} : \langle \mathsf{Test} \rangle; \mathsf{end}\}.$

Figure 2: Isomorphic eHealth Protocol (branch and prefix swapping combinators applied)

Consider communication between participants p and q, where p sends choices, identified with labels $l_i, i \in I$, to proceed within each branch exchanging value U between participants p$'$ and q$'$ and then continue as global type G_i branch. Then these two communications can be swapped reflecting concurrency of the interaction:

$$g_1; g; G_1 \times \ldots \times g_i; g; G_i \overset{\text{Contr}}{\underset{\text{Exp}}{\rightleftarrows}} g; (g_1; G_1 \times \ldots \times g_i; G_{i,i\in I}) \qquad \textbf{(Branching)}$$

$$\mathsf{Contr}(G) \triangleq \lambda g \lambda g_1 \ldots \lambda g_k \lambda G_1 \ldots \lambda G_k. \quad \text{if} \quad G = g_1; g; G_1 \times \ldots \times g_k; g; G_k \quad \text{and}$$
$$\mathsf{pid}(g) \cap \mathsf{pid}(g_i) = \emptyset, 1 \le i \le k \quad \text{then} \quad g; (g_1; G_1 \times \ldots \times g_k; G_k) \quad \text{else} \quad G. \qquad (3)$$

The inverse of the contracting function Contr will be expanding one:

$$\mathsf{Exp}(G) \triangleq \lambda g \lambda g_1 \ldots \lambda g_k \lambda G_1 \ldots \lambda G_k. \quad \text{if} \quad G = g; (g_1; G_1 \times \ldots \times g_k; G_k) \quad \text{and}$$
$$\mathsf{pid}(g) \cap \mathsf{pid}(g_i) = \emptyset, 1 \le i \le k \quad \text{then} \quad g_1; g; G_1 \times \ldots \times g_k; g; G_k \quad \text{else} \quad G. \qquad (4)$$

The next swapping equivalence for the global type is the analogue of the distributivity for branching within branches (indexed prefixes reflect the labels exchanged):

$$g_1; (\overline{g}_{n+1}; G_1 \times \ldots \times \overline{g}_{n+k}; G_k) \times \ldots \times g_n; (\overline{g}_{n+1}; G_1 \times \ldots \times \overline{g}_{n+k}; G_k) \overset{\text{SwapBr}_l}{\underset{\text{SwapBr}_r}{\rightleftarrows}}$$

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

In the case of the eHealth logging protocol, as we mentioned, there is an independent communication between Insurance and the Records system within the choice sent from the Doctor to the Patient. Applying contracting combinator Contr to the G_{sw} isomorphic to the G, we arrive to another isomorphic protocol G_{sw}^{br} depicted in Fig.2. Projections of this transformation G_{sw}^{br} of the original protocol G onto the Patient, Doctor and Insurance participant are exactly the same. An interesting case arises when looking at the local type for the Hospital Record: syntactically projections on this participant of the global types from the Fig.1 and Fig.2 are different. However trace sets of the projections are equivalent $Tr(\mathsf{G} \upharpoonright \mathsf{R}) = Tr(\mathsf{G}_{sw}^{br} \upharpoonright \mathsf{R}).$

3   Semantics of Multiparty Session Type Isomorphism

The common framework to describe session networks is to study configurations. Configurations are collections of *local types* corresponding to remaining expected actions for all participants. We follow notation introduced in [6] to compare sets of languages of local message traces for associated global types.

Definition 5 *(Configuration Traces) A configuration trace σ is a mapping from participants to a sequence of labels of local types, i.e. $\sigma(r) = \ell_1...\ell_n$ where $\ell_i \in \mathfrak{L}$. A participant r is in the domain of σ if $\sigma(r) \neq \varepsilon$ where ε stands for an empty sequence.*

Definition 6 *(Configurations) Given a set of roles $\mathscr{P}$, we define a configuration as $\Delta = (T_p)_{p \in \mathscr{P}}$ where T_p is a local type projected to participant p (i.e. a local type of participant p). The synchronous transition relation between configurations is defined as:*

$$\frac{T_p \xrightarrow{\ell} T_p' \quad T_q \xrightarrow{\overline{\ell}} T_q' \quad T_r = T_r' \quad r \neq p, r \neq q}{(T_p)_{p \in \mathscr{P}} \xrightarrow{\ell \cdot \overline{\ell}} (T_p')_{p \in \mathscr{P}}} \text{ (SYNCH)}$$

The relation between traces and configuration is given by execution relation $\Delta \rightsquigarrow^{\sigma}_{\text{synch}} \Delta'$:

Definition 7 *(Configuration Execution and Traces) Configuration Δ executes trace σ to configuration Δ' for synchronous semantics, if*

1. *For any configuration Δ, $\Delta \rightsquigarrow^{\sigma_0}_{\text{synch}} \Delta$, where $\forall r : \sigma_0(r) = \varepsilon$*

2. *For any configurations $\Delta, \Delta_1, \Delta_2$ any trace σ, any label l, if $\Delta \rightsquigarrow^{\sigma}_{\text{synch}} \Delta_1$ and $\Delta_1 \xrightarrow{\ell \cdot \overline{\ell}} \Delta_2$ within synchronous semantics, then we define $\Delta \rightsquigarrow^{\sigma'}_{\text{synch}} \Delta_2$ as follows:*

    $\ell \cdot \overline{\ell} = q!m \cdot p?m, \quad \text{then} \quad \sigma'(p) = \sigma(p).q!m, \sigma'(q) = \sigma(q).p?m \quad \text{and} \quad \sigma'(r) = \sigma(r), r \notin \{p, q\}.$

Definition 8 *(Denotation of a Global Type and Terminated Traces). Let us define $\delta(G) = (T_p)_{p \in \mathscr{P}}$ where $\mathscr{P}$ is a set of participants in G. We define the denotation of global type G under synchronous semantic, denoted $D(G)$, 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 $\Delta \not\rightarrow$.*

Therefore we arrive to the statement about global type isomorphism with relation to the local traces: isomorphic global types will have the same sets of traces. We can show for the three isomorphisms (Commutativity, Branching and Branching distributivity), that two isomorphic types will have equal trace sets, i.e. executing the same action (communication between two participants) on each isomorphic global type will result in equivalent up to defined isomorphism global types.

Lemma 1 *If $G_1 \rightleftarrows G_2$, then $Tr(G_1) = Tr(G_2)$.*

We prove Lemma 1 by induction over global operational semantics given in Table 3 for the three isomorphism rules. The proof is given in Appendix A.

Next Theorem 1 shows the equivalence between trace sets of a global type and configuration traces of a set of local types projected from that global type. Let us denote the trace set of the configuration of the global type by $\mathscr{T}_S(\Delta)$. The following theorem proves the trace set of the global type G for the synchronous semantic, $Tr(G)$, is equivalent to $\mathscr{T}_S(\Delta)$. Below the equivalence relation $\equiv$ is defined by identifying $g = p \rightarrow q : m$ (the label of the global type trace) to $q!m \cdot p?m$ (the labels of the configuration trace). The proof is given in Appendix B.

Theorem 1 (Equivalence between Synchronous Global Types and Configuration Traces) *Let G be a global type with participants $\mathscr{P}$ and let $\Delta = (G \restriction \mathrm{p})_{\mathrm{p} \in \mathscr{P}}$ be the local type configuration projected from G. Then $Tr(G) \equiv \mathscr{T}_S(\Delta)$ where $\Delta = (T_{\mathrm{p}})_{\mathrm{p} \in \mathscr{P}}$.*

By Lemma 1 and Theorem 1, Theorem 2 concludes that the denotational semantics of two isomorphic global types are the same.

Theorem 2 (Soundness) *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 $T_{i\mathrm{p}} = G_i \restriction \mathrm{p}$. Hence if $G_1 \rightleftarrows G_2$, then $\boldsymbol{D}(G_1) = \boldsymbol{D}(G_2)$.*

We conjecture the completeness direction.

4   Related Work and Conclusions

Type Isomorphism   The main theory of *type isomorphisms* developed in [9] demonstrated that type theory is an effective formalism to classify software components and how type isomorphism can be practically employed to catalogue and manage behaviorally equivalent components. However, isomorphisms are often considered to be too strict in distributed settings, whose behavioural semantics is often given by means of process calculus. The need for the latter formalism of distributed component equivalence was a historical motivation for developing notions of component similarity and adaptation via bisimulation [12], testing equivalences [1] and so on.

In contrast, our pursuit of defining isomorphism framework for the globally governed semantic of multiparty processes is an attempt to find more flexible type level equalities. We build upon earlier work to axiomatise session type isomorphisms through behavioral adaptation. The first such attempt to investigate session type isomorphisms, following the theory of type isomorphisms [5] and finite hereditary permutations, was presented in [8] and described combinators for *binary session types isomorphisms* corresponding to adjacent processes. Interpretation of linear logic propositions as session types for communicating processes explains how type isomorphisms resulting from linear logic equivalences are realised by coercions between interface types of session-based concurrent systems [13].

We extend our investigation beyond binary session types to multiparty session types (MPST) [10].

Global Protocol Adaptation   Works addressing adaptation for multiparty communications include [14], [4] and [2] . The paper [14] proposes a choreographic language for distributed applications. Adaptation follows a rule-based approach, in which all interactions, under all possible changes produced by the adaptation rules, proceed as prescribed by an abstract model. In [4] a calculus based on global types, monitors and processes is introduced and adaptation is triggered after the execution of the communications prescribed by a global type, in reaction to changes of the global state. In contrast, in [2] adaptation is triggered by security violations, and assures access control and secure information flow.

Trace Semantics for MPST   The first study of the expressiveness of multiparty session types through trace semantics is given in [6]. That work employs sets of languages of local message traces to compare expressiveness of different semantics of multiparty session types based on: the presence and nature of varied data structures (input or output queues), flexibility of the local types, defined as a subtyping relation, and presence of parallel sessions and interruptions. The global type isomorphism design we offer here is straightforwardly extendable to this semantics.

Our future work includes the extensions to subtyping, sessions with interruptions, and asynchronous semantics. At the practical side, it is interesting to implement combinators in functional languages such as Haskell or OCaml taking Scribble [15] as a source global protocol.

5   Acknowledgements

References

[1] Giovanni Bernardi & Matthew Hennessy (2015): *Mutually Testing Processes.* *LMCS* 11(2), doi:10.2168/LMCS-11(2:1)2015.

[2] Ilaria Castellani, Mariangiola Dezani-Ciancaglini & Jorge A. Pérez (2016): *Self-adaptation and secure information flow in multiparty communications.* *Formal Asp. Comput.* 28(4), pp. 669–696, doi:10.1007/s00165-016-0381-3.

[3] Mario Coppo, Mariangiola Dezani-Ciancaglini, Luca Padovani & Nobuko Yoshida (2015): *A Gentle Introduction to Multiparty Asynchronous Session Types.* In: *SFM 2015*, pp. 146–178, doi:10.1007/978-3-319-18941-3_4.

[4] Mario Coppo, Mariangiola Dezani-Ciancaglini & Betti Venneri (2015): *Self-adaptive multiparty sessions.* *Service Oriented Computing and Applications* 9(3-4), pp. 249–268, doi:10.1007/s11761-014-0171-9.

[5] Roberto Di Cosmo (2005): *A short survey of isomorphisms of types.* *MSCS* 15(5), pp. 825–838, doi:10.1017/S0960129505004871.

[6] Romain Demangeon & Nobuko Yoshida (2015): *On the Expressiveness of Multiparty Sessions.* In: *FSTTCS 2015*, *LIPIcs* 45, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 560–574, doi:10.4230/LIPIcs.FSTTCS.2015.560.

[7] Pierre-Malo Deniélou & Nobuko Yoshida (2013): *Multiparty Compatibility in Communicating Automata: Characterisation and Synthesis of Global Session Types.* In: *ICALP 2013*, pp. 174–186, doi:10.1007/978-3-642-39212-2_18.

[8] Mariangiola Dezani-Ciancaglini, Luca Padovani & Jovanka Pantovic (2014): *Session Type Isomorphisms.* In: *PLACES 2014*, *EPTCS* 155, pp. 61–71, doi:10.4204/EPTCS.155.9.

[9] Roberto Di Cosmo (1995): *Isomorphisms of types: from λ-calculus to information retrieval and language design.* Birkhauser, doi:10.1007/978-1-4612-2572-0.

[10] Kohei Honda, Nobuko Yoshida & Marco Carbone (2008): *Multiparty asynchronous session types.* In: *POPL2008*, pp. 273–284, doi:10.1145/1328438.1328472.

[11] Dimitrios Kouzapas & Nobuko Yoshida (2014): *Globally Governed Session Semantics.* *Logical Methods in Computer Science* 10(4), doi:10.2168/LMCS-10(4:20)2014.

[12] Robin Milner (1984): *Lectures on a Calculus for Communicating Systems.* In: *Seminar on Concurrency, Carnegie-Mellon University, Pittsburg, PA, USA, July 9-11, 1984*, pp. 197–220, doi:10.1007/3-540-15670-4_10.

[13] Jorge A. Pérez, Luís Caires, Frank Pfenning & Bernardo Toninho (2014): *Linear logical relations and observational equivalences for session-based concurrency.* *Inf. Comput.* 239, pp. 254–302, doi:10.1016/j.ic.2014.08.001.

[14] Mila Dalla Preda, Maurizio Gabbrielli, Saverio Giallorenzo, Ivan Lanese & Jacopo Mauro (2017): *Dynamic Choreographies: Theory And Implementation.* *Logical Methods in Computer Science* 13(2), doi:10.23638/LMCS-13(2:1)2017.

[15] Nobuko Yoshida, Raymond Hu, Rumyana Neykova & Nicholas Ng (2013): *The Scribble Protocol Language*. In: *Trustworthy Global Computing - 8th International Symposium, TGC 2013*, pp. 22–41, doi:10.1007/978-3-319-05119-2_3.

A   Proof of Lemma 1

Recall Definition 3 of the trace set of the global type. We will show for the three isomorphisms, that two isomorphic types will have the same trace sets, i.e. executing the same action(communication between two participants) on each isomorphic global type will result in the same or equivalent up to isomorphism global type. We will start with the *prefix swapping* isomorphism, that reflect the reordering of message passing for the pairwise different participants.

- $\underbrace{g_1;g_2;G}_{G_1} \rightleftarrows \underbrace{g_2;g_1;G}_{G_2}$

  Proof Let us recall operational semantics Table 3 for the Global types progress. There are three possibilities for the types G_1 and G_2 to proceed:

  1. Execute communication g_1:

     Following the rule [Inter], type G_1 will reduce to: $g_1;g_2 : G \xrightarrow{g_1} g_2;G$. To execute the same trace on G_2, we apply rule [IPerm]:$g_2;g_1 : G \xrightarrow{g_1} g_2;G$. Both times execution of this trace results in the same global type $g_2;G$, hence $Tr(G_1) = Tr(G_2)$.

  2. Execute communication between participants in g_2:

     By the rule [Iperm] global type G_1 reduces to $g_1;g_2;G \xrightarrow{g_2} g_1;G$. The same trace on the global type G_2 will utilise [Inter] operational rule: $g_2;g_1 : G \xrightarrow{g_2} g_1;G$. Both G_1 and G_2 reduce to the same global type $g_1;G$, which implies trace equality on this path: $Tr(G_1) = Tr(G_2)$.

  3. Execute communication g within global type G between participants that are not involved in g_1 and g_2, reducing G into G':

     $$\frac{G \xrightarrow{g} G' \qquad \mathsf{pid}(g_1) \cap \mathsf{pid}(g_2) \cap \mathsf{pid}(g) = \emptyset}{G_1 = g_1;g_2;G \xrightarrow{g} g_1;g_2;G' \qquad G_2 = g_2;g_1;G \xrightarrow{g} g_2;g_1;G'} \text{ [IPerm]}$$

     We arrive to the global types, equivalent up to prefix swapping isomorphism, by induction step on $g_1;g_2;G'$ and $g_2;g_1;G'$, therefore $Tr(G_1) = Tr(G_2)$.

  Thus, these two suspect isomorphic global types G_1 and G_2 have the same trace sets and by our definition of the global multiparty session type isomorphism are indeed isomorphic. ∎

- $\underbrace{g_1;g;G_1 \times \ldots \times g_i;g;G_i}_{\overline{G_1}} \rightleftarrows \underbrace{g;(g_1;G_1 \times \ldots \times g_i;G_{i,i\in I})}_{\overline{G_2}}$

  Proof Following operational semantic for the global types, we distinguish three cases of the trace execution:

  1. Execute communication between participants offering branching selection in the k-th branch: $g_1;g;G_1 \times \ldots \times g_i;g;G_i \xrightarrow{g_k} g;G_k$ by the rule [SelBra]. The same trace on the global type $\overline{G_2}$ will require application of the [IPerm] rule: $\dfrac{g_1;G_1 \times \ldots \times g_i;G_{i,i\in I} \xrightarrow{g_k} G_k \quad \text{[SelBra]}}{g;(g_1;G_1 \times \ldots \times g_i;G_{i,i\in I}) \xrightarrow{g_k} g;G_k} \text{ [IPerm]}$

2. Executing communication within prefix g

$$\frac{g; G_k \xrightarrow{g} G_k, \quad \mathsf{pid}(g) \cap \mathsf{pid}(g_i) = \emptyset, \ k \in I \quad [\mathsf{Inter}]}{g_1; g; G_1 \times \ldots \times g_i; g; G_i \xrightarrow{g} g_1; G_1 \times \ldots \times g_i; G_{i,i \in I}} [\mathsf{SBPerm}]$$

while $\overline{G_2} \xrightarrow{g} G_0$ when applying rule [Inter].

3. Executing communication $\overline{g}$ that moves global types G_i into corresponding $\overline{G_i}$ is similar to the case 2 above.

B Proof of the Theorem 1

Proof By induction on reduction of the global type LTS we show that if $\delta(G) \rightsquigarrow^{\sigma}_{\mathsf{synch}} \delta(G')$, then $\mathscr{T}(\Delta) \rightsquigarrow^{\sigma}_{\mathsf{synch}} \mathscr{T}(\Delta')$, i.e. if the trace σ is in the trace set of a global type G, then it is also in the trace set of the configuration corresponding to this global type.

- Let $G = \mathsf{end}$. This is a trivial case and $\delta(\mathsf{end}) \equiv \mathscr{T}(\mathsf{end}) = \varepsilon$.

- Let $G = g; G_1$, where $g = \mathsf{p} \rightarrow \mathsf{q} : \langle U \rangle$ and $\mathsf{pid}(G_1) = \{\mathsf{r}_1, ..\mathsf{r}_n\} = \overline{\mathsf{r}}$

  Then configuration Δ of the global type G is the set of its local projections, $\Delta = T_\mathsf{p}, T_\mathsf{q}, T_{\overline{\mathsf{r}}}$. There are two possibilities for the global type G to proceed according to the operational semantics of LTS in Table 3:

  1. $G \xrightarrow{g} G'$ Then configuration of the global type G will follow the transition relation [Synch] rule with the trace $\sigma = \mathsf{inp}(g)!\langle U \rangle \cdot \mathsf{out}(g)?\langle U \rangle$:

  $$\Delta = \mathsf{inp}(g)!\langle U \rangle; T_{\mathsf{out}(g)}, \mathsf{out}(g)?\langle U \rangle; T_{\mathsf{inp}(g)}, T_{\overline{\mathsf{r}}} \xrightarrow{\mathsf{inp}(g)!\langle U \rangle \cdot \mathsf{out}(g)?\langle U \rangle} T_{\mathsf{out}(g)}, T_{\mathsf{inp}(g)}, T_{\overline{\mathsf{r}}} = \Delta'$$

  At the same time $\delta_S(G') : T_{\mathsf{out}(g)}, T_{\mathsf{inp}(g)}, T_{\overline{\mathsf{r}}}$.

  2. $G \xrightarrow{g'} G'$, where $g' = \mathsf{r} \rightarrow \mathsf{s} : \langle U' \rangle$ and $\mathsf{empty}_S(g, g')$, the trace we execute in this case will be $\sigma = \mathsf{inp}(g')!\langle U' \rangle \cdot \mathsf{out}(g')?\langle U' \rangle$

  $$\Delta = \mathsf{inp}(g)!\langle U' \rangle; T_{\mathsf{out}(g)}, \mathsf{out}(g)?\langle U' \rangle; T_{\mathsf{inp}(g)}, \mathsf{inp}(g')!\langle U' \rangle; T_{\mathsf{out}(g')}, \mathsf{out}(g')?\langle U' \rangle; T_{\mathsf{inp}(g')} T_{\overline{\mathsf{r}}} \rightarrow$$
  $$\xrightarrow{\mathsf{inp}(g')!\langle U' \rangle \cdot \mathsf{out}(g')?\langle U' \rangle} \mathsf{inp}(g)!\langle U' \rangle; T_{\mathsf{out}(g)}, \mathsf{out}(g)?\langle U' \rangle; T_{\mathsf{inp}(g)}, T_{\mathsf{out}(g')}, T_{\mathsf{inp}(g')}, T_{\overline{\mathsf{r}}} = \Delta' \quad (5)$$

  Transition of the configuration Δ reduces to the configuration Δ', meanwhile trace of the reduced global type $\delta(G') = \Delta'$

- Let $G = g_1; G_1 \times \ldots \times g_k; G_{k, k \in I}$, where $g_i = \mathsf{p} \rightarrow \mathsf{q} : l_i, i \in I$ and $\mathsf{pid}(G_i) = \overline{\mathsf{r}}_i$. There are two cases depending on the transition rule used for the global type operational semantics:

  1. $G \xrightarrow{g_i} G_i$ following [SelBra] rule. Corresponding global type configuration Δ will be executing trace $\sigma = \mathsf{inp}(g_i)!l_i \cdot \mathsf{out}(g_i)?l_i$.

  2. $G \xrightarrow{g'_i} G'$ using [SBPerm] rule. Corresponding configuration will be executing trace $\sigma = \mathsf{inp}(g')!l'_i \cdot \mathsf{out}(g')?l'_i$. In both cases configuration trace of the branching global type will be coinciding with reduced global type configuration.

■