

# Multiparty Session Types, Beyond Duality (Abstract)

Alceste Scalas

Imperial College London  
alceste.scalas@imperial.ac.uk

Nobuko Yoshida

Imperial College London  
n.yoshida@imperial.ac.uk

Multiparty Session Types (MPST) are a well-established typing discipline for message-passing processes interacting on *sessions* involving two or more participants. Session typing can ensure desirable properties: absence of communication errors and deadlocks, and protocol conformance. However, existing MPST works provide a *subject reduction* result that is arguably (and sometimes, surprisingly) restrictive: it only holds for typing contexts with strong *duality* constraints on the interactions between pairs of participants. Consequently, many “intuitively correct” examples cannot be typed and/or cannot be proved type-safe. We illustrate some of these examples, and discuss the reason for these limitations. Then, we outline a novel MPST typing system that removes these restrictions.

**MPST in a Nutshell** In the MPST framework [4], *global types* (describing interactions among *roles*) are projected to *local types* used to type-check *processes*. E.g., the global type  $G$  involves roles  $p, q, r$ :

$$G = p \rightarrow q: \left\{ \begin{array}{l} m1(\text{Int}).q \rightarrow r:m2(\text{Str}).r \rightarrow p:m3(\text{Bool}).\mathbf{end}, \\ \text{stop}.q \rightarrow r:\mathbf{quit}.\mathbf{end} \end{array} \right\}$$

$G$  says that  $p$  sends to  $q$  *either* a message  $m1$  (carrying an  $\text{Int}$ ) *or*  $\text{stop}$ ; in the first case,  $q$  sends  $m2$  to  $r$  (carrying a  $\text{Str}$ ), then  $r$  sends  $m3$  to  $p$  (carrying a  $\text{Bool}$ ), and the session **ends**; otherwise, in the second case,  $q$  sends  $\mathbf{quit}$  to  $r$ , and the session **ends**. The *projections of  $G$*  are the I/O actions of each role in  $G$ :

$$S_p = q \oplus \left\{ \begin{array}{l} m1(\text{Int}).r \oplus m3(\text{Bool}), \\ \text{stop} \end{array} \right\} \quad S_q = p \& \left\{ \begin{array}{l} m1(\text{Int}).r \oplus m2(\text{Str}), \\ \text{stop}.r \oplus \mathbf{quit} \end{array} \right\} \quad S_r = q \& \left\{ \begin{array}{l} m2(\text{Str}).p \oplus m3(\text{Bool}), \\ \mathbf{quit} \end{array} \right\}$$

Here,  $S_p, S_q, S_r$  are the projections of  $G$  resp. onto  $p, q, r$ . E.g.,  $S_p$  is a session type that represents the behaviour of  $p$  in  $G$ : it must send ( $\oplus$ ) to  $q$  either  $m1(\text{Int})$  or  $\text{stop}$ ; in the first case, the channel is then used to receive ( $\&$ ) message  $m3(\text{Bool})$  from  $r$ , and the session ends; otherwise, in the second case, the session ends. Now, a *typing context*  $\Gamma$  can assign types  $S_p, S_q$  and  $S_r$  to *multiparty channels*  $s[p], s[q]$  and  $s[r]$ , used to play roles  $p, q$  and  $r$  on *session*  $s$ . Then, if e.g. some parallel processes  $P_p, P_q$  and  $P_r$  type-check w.r.t.  $\Gamma$ , then we know that such processes use the channels abiding by their types.

**Subject Reduction, or Lack Thereof** We would expect that typed processes reduce type-safely, e.g.:

$$\vdash P \triangleright \Gamma \text{ and } P \rightarrow^* P' \text{ implies } \exists \Gamma' : \vdash P' \triangleright \Gamma' \quad (\text{where } P = P_p | P_q | P_r \text{ and } \Gamma = s[p]:S_p, s[q]:S_q, s[r]:S_r) \quad (1)$$

But surprisingly, *this is not the case!* In MPST works (e.g., [1]), the subject reduction statement reads:

$$\vdash P \triangleright \Gamma \text{ with } \Gamma \text{ consistent and } P \rightarrow^* P' \text{ implies } \exists \Gamma' \text{ consistent such that } \vdash P' \triangleright \Gamma' \quad (2)$$

Intuitively,  $\Gamma$  is consistent if all its potential interactions between pairs of roles are *dual*: e.g., all potential outputs of  $S_p$  towards  $r$  are matched by compatible input capabilities of  $S_r$  from  $p$ . Consistency

is quite restrictive, due to its (rather intricate) *syntactic* nature—and does *not* hold in our example. This is due to *inter-role dependencies*:  $S_p$  allows to decide what to send to  $q$  — and depending on such a choice, whether to input  $m3$  from  $r$ , or not. This breaks the definition of consistency between  $S_p$  and  $S_r$ ; hence,  $\Gamma$  in (1) is not consistent, and we cannot apply (2) to ensure that  $P_p, P_q, P_r$  reduce type-safely.

**Our Proposal** In “standard” MPST works, consistency cannot be lifted without breaking subject reduction [1, p.163]. Hence, to prove that our example is type-safe, we need to revise the MPST foundations. We propose a *novel MPST typing system* that safely lifts the consistency requirement, by introducing:

1. a new MPST typing judgement with the form  $\Theta \vdash P \triangleright \Gamma_g \triangleleft \Gamma_r$  —where  $\Gamma_g$  and  $\Gamma_r$  are respectively the *guarantee* and *rely typing contexts*. Intuitively,  $\Gamma_g$  describes how  $P$  uses its channels, while  $\Gamma_r$  describes how other processes (possibly interacting with  $P$ ) are expected to use their channels;
2. a *semantic* notion of typing context safety, called *liveness*, based on MPST context reductions [1]. In our typing judgement, the pair  $\Gamma_g, \Gamma_r$  must be live: this ensures that each output can synchronise with a compatible input (and *vice versa*). Unlike consistency, liveness supports complex inter-role dependencies, and ensures that the typing context cannot deadlock.

**Related Work** A technical report with more examples and discussion is available in [6]. Our novel typing system allows to prove type safety of processes implementing global types with complex inter-role dependencies and delegations. To the best of our knowledge, the only work with a similar capability is [3]; however, its process calculus only supports *one* session, and this restriction is crucially exploited to type parallel compositions without “splitting” them (cf. Table 8, rule [T-SESS]). Hence, unlike our work, [3] does not support multiple sessions and delegation—and extending it seems challenging. Further, unlike [3], our typing rules do *not* depend on global types and projections: by removing this orthogonal concern, we simplify the theory. If needed, a set of local types can be related to a global type via “top-down” projection or “bottom-up” synthesis [5]. Similarly to most MPST papers, our work ensures that a typed process (*vs*)  $(\prod_{p \in I} P_p)$ , with each  $P_p$  only interacting on  $s[p]$ , is deadlock-free—but does not guarantee deadlock freedom for multiple interleaved sessions [2]: we leave this topic as future work.

**Thanks** to the reviewers for their suggestions, and to R. Hu, J. Lange, B. Toninho for the fruitful discussion. Work supported by: EPSRC (EP/K011715/1, EP/K034413/1, EP/L00058X/1), EU (COST Action IC1201, FP7-612985).

## References

- [1] M. Coppo, M. Dezani-Ciancaglini, L. Padovani & N. Yoshida (2015): *A Gentle Introduction to Multiparty Asynchronous Session Types*. doi:10.1007/978-3-319-18941-3\_4.
- [2] M. Coppo, M. Dezani-Ciancaglini, N. Yoshida & L. Padovani (2016): *Global Progress for Dynamically Interleaved Multiparty Sessions*. *MSCS* 26(2), doi:10.1017/S0960129514000188.
- [3] M. Dezani-Ciancaglini, S. Ghilezan, S. Jakšić, J. Pantović & N. Yoshida (2016): *Precise subtyping for synchronous multiparty sessions*. In: *PLACES 2015*, doi:10.4204/EPTCS.203.3.
- [4] K. Honda, N. Yoshida & M. Carbone (2008): *Multiparty asynchronous session types*. In: *POPL*, doi:10.1145/1328438.1328472. Full version: Volume 63, Issue 1, March 2016 (9), pages 1-67, *JACM*.
- [5] J. Lange, E. Tuosto & N. Yoshida (2015): *From Communicating Machines to Graphical Choreographies*. In: *POPL*, doi:10.1145/2676726.2676964.
- [6] A. Scalas & N. Yoshida (2017): *Multiparty Session Types, Beyond Duality*. Technical Report, Imperial College London. Available at <https://www.doc.ic.ac.uk/research/technicalreports/2017/>.