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 $\mathtt{p}$, $\mathtt{q}$, $\mathtt{r}$:

$$G = \mathtt{p}{\to}\mathtt{q}{:}\left\{\begin{matrix}\mathtt{m1}(\mathrm{Int})\,.\,\mathtt{q}{\to}\mathtt{r}{:}\mathtt{m2}(\mathrm{Str})\,.\,\mathtt{r}{\to}\mathtt{p}{:}\mathtt{m3}(\mathrm{Bool})\,.\,\mathbf{end}\,,\\ \mathtt{stop}\,.\,\mathtt{q}{\to}\mathtt{r}{:}\mathtt{quit}\,.\,\mathbf{end}\end{matrix}\right\}$$

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

$$S_{\mathtt{p}} = \mathtt{q}\bigoplus\left\{\begin{matrix}\mathtt{m1}(\mathrm{Int})\,.\,\mathtt{r}\&\mathtt{m3}(\mathrm{Bool})\,,\\ \mathtt{stop}\end{matrix}\right\} \quad S_{\mathtt{q}} = \mathtt{p}\,\&\left\{\begin{matrix}\mathtt{m1}(\mathrm{Int})\,.\,\mathtt{r}\oplus\mathtt{m2}(\mathrm{Str})\,,\\ \mathtt{stop}\,.\,\mathtt{r}\oplus\mathtt{quit}\end{matrix}\right\} \quad S_{\mathtt{r}} = \mathtt{q}\,\&\left\{\begin{matrix}\mathtt{m2}(\mathrm{Str})\,.\,\mathtt{p}\oplus\mathtt{m3}(\mathrm{Bool})\,,\\ \mathtt{quit}\end{matrix}\right\}$$

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

$$\vdash P \rhd \Gamma \text{ and } P \to^* P' \text{ implies } \exists\Gamma' : \vdash P' \rhd \Gamma' \quad (\text{where } P = P_{\mathtt{p}} \,|\, P_{\mathtt{q}} \,|\, P_{\mathtt{r}} \text{ and } \Gamma = s[\mathtt{p}]{:}S_{\mathtt{p}}, s[\mathtt{q}]{:}S_{\mathtt{q}}, s[\mathtt{r}]{:}S_{\mathtt{r}}) \quad (1)$$

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

$$\vdash P \rhd \Gamma \ \underline{\textit{with } \Gamma \textit{ consistent}} \text{ and } P \to^* P' \text{ implies } \exists\Gamma' \ \underline{\textit{consistent}} \text{ such that } \vdash P' \rhd \Gamma' \quad (2)$$

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

To appear in EPTCS.

is quite restrictive, due to its (rather intricate) *syntactic* nature—and does *not* hold in our example. This is due to *inter-role dependencies*: $S_{\mathtt{p}}$ allows to decide what to send to $\mathtt{q}$ — and depending on such a choice, whether to input $\mathtt{m3}$ from $\mathtt{r}$, or not. This breaks the definition of consistency between $S_{\mathtt{p}}$ and $S_{\mathtt{r}}$; hence, Γ in (1) is not consistent, and we cannot apply (2) to ensure that $P_{\mathtt{p}}$, $P_{\mathtt{q}}$, $P_{\mathtt{r}}$ 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 Γ_g and Γ_r are respectively the *guarantee* and *rely typing contexts*. Intuitively, Γ_g describes how P uses its channels, while Γ_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 Γ_g, Γ_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 $(\nu s)\,(|_{\mathtt{p}\in I} P_{\mathtt{p}})$, with each $P_{\mathtt{p}}$ only interacting on $s[\mathtt{p}]$, 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/`.