Expressiveness, Separation and Complexity Analyses on Multiparty Session Types

Nobuko Yoshida
University of Oxford, Oxford, UK
nobuko.yoshida@cs.ox.ac.uk

Multiparty session types (MPST) are a type discipline for concurrent and distributed systems that ensures safety, deadlock freedom, and liveness of communicating components. This paper surveys recent advances in the *expressiveness*, *separation*, and *complexity* of MPST, with a particular focus on recent results in [63, 74, 35, 65, 52].

1 Introduction

Multiparty session types (MPST) [33, 32] are a type discipline for designing and verifying communication protocols involving multiple participants in concurrent and distributed systems. This paper discusses the *expressiveness* of the MPST framework.

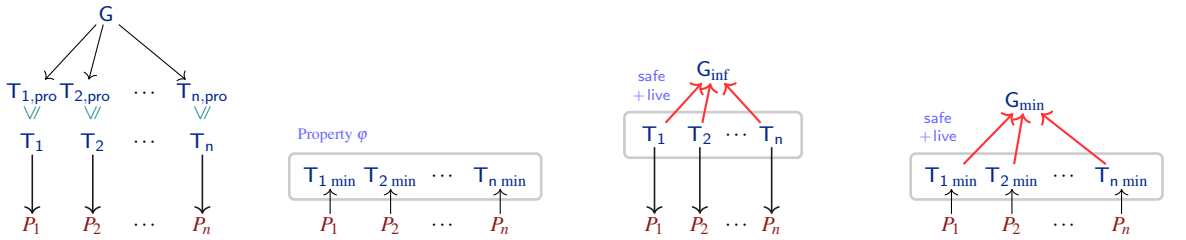
In the EXPRESS community, the term "expressiveness" usually means "*encodability*" between process calculi [27]. An *encodability* result establishes an encoding from a *source calculus* to a *target calculus* that satisfies a given set of criteria, showing that the target can emulate the behaviour of the source and is therefore at least as expressive. Conversely, a *separation* result proves that *no such encoding exists*, demonstrating that certain behaviours of the source cannot be enumerated by the target. Encodability and separation results establish expressiveness hierarchies among calculi, provided they are based on the same correctness criteria.

Peters and Yoshida [63] have proved that a synchronous *mixed choice multiparty session calculus*, typed using the bottom-up strategy (see the next section), is strictly more expressive than multiparty session calculi with only *separate choice*. Here, *mixed choice* allows one choice to contain both input and output actions, but *separate choice* contains either input or output actions in the same summand. The same work also established that the *binary* mixed session calculus [6] is strictly less expressive, separating the expressive power of binary and multiparty session types.

More recently, Masters and Yoshida [52] have proved that the subtyping (refinement) relation of [63] is *precise*: the subtyping rules accept *exactly* the safe subtype relation while rejecting every unsafe one. Consequently, the typing system of [63] achieves the maximum expressiveness among synchronous mixed choice session calculi. But what does this notion of "maximum expressiveness" actually mean?

The remainder of this paper discusses expressiveness from a different perspective: the expressive power of *typing systems* with subtyping relations. Here, expressiveness refers to the *typability* of processes under two different typing systems with the same syntax and operational semantics. If the two typing systems yield exactly the same set of typable processes, they are said to have the same expressive power.

Submitted to:
EXPRESS/SOS 2026



(a) top-down (b) bottom-up (c) global type inference (d) principal type inference

Figure 1: Multiparty session type frameworks from [65, Fig.1]

2 Expressiveness as Typability

This section discusses the expressiveness of two typing systems, namely the top-down and bottom-up typing systems.

2.1 Top-Down Strategy vs Bottom-Up Strategy

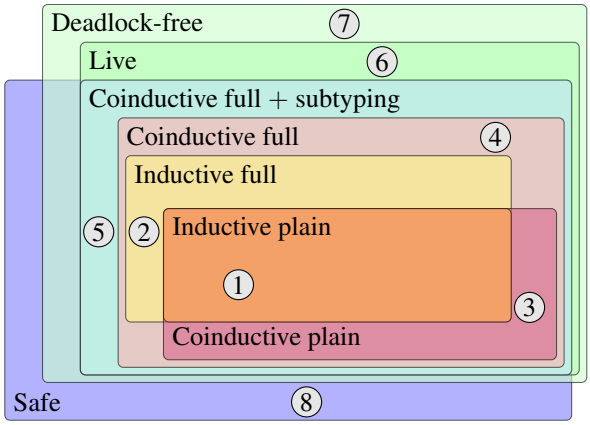
Top-Down Strategy. The original MPST theory [32, 33] takes the *top-down* strategy (Figure 1a). Instead of developing multiple distributed programs independently and hoping that their communications match correctly, a protocol designer first specifies the overall communication protocol—who sends what to whom, and with what type—as a *global type* G. An *endpoint projection* (EPP) algorithm then generates a set of local types $\{\mathsf{T}_{i,\mathsf{pro}}\}_{i\in I}$, each prescribing how an individual program P_i may communicate with the others.

Type checking guarantees that the resulting processes $\{P_i\}_{i\in I}$ satisfy *communication safety* (no unexpected type or label mismatches), *session fidelity* (each process conforms to the specified global protocol), and *deadlock freedom*. In the top-down approach, typability also guarantees the stronger property of *liveness*, which means that every request eventually receives a response and that each participant can continue to make progress without stalling indefinitely.

The top-down strategy has been integrated into numerous programming languages and software frameworks through code generation, API generation and inference, event-driven programming, static type checking, protocol compliance, real-time systems, and runtime monitoring. Representative implementations include Rust [13, 46, 34, 75], Java [36, 37, 45, 4], Scala [66, 76, 12, 2, 19, 41], Go [9], TypeScript [54, 23], PureScript [44], OCaml [78, 39], MPI-C [59, 51], F⋆ [79], F♯ [55], Python [58, 57, 14], Erlang [20, 56, 15, 3, 22], C [61, 62], C♯ [43], and domain-specific languages [48, 10, 11, 28, 21]. A comprehensive survey is given in [77].

The correctness of the top-down framework has also been verified in several theorem-proving frameworks [7, 72, 73, 40, 16, 49, 8, 42].

Bottom-Up Strategy. The bottom-up strategy, proposed in [67], directly verifies that a collection of local types–either written by the programmer or inferred from a set of programs–satisfies a desired property φ (Figure 1b). The motivation of [67] was to develop a general MPST framework that does not rely on prescribing a global type, thereby establishing *type soundness* without relying on global types or endpoint projection (EPP). The paper further argued that existing proofs of type soundness based on endpoint projection with the *mergeability* operator are flawed. Since mergeability enlarges the class of



	Complexity
①	$\Theta(\lvert\mathsf{G}\rvert \log \lvert\mathsf{G}\rvert)$
②	$\Theta(\lvert\mathsf{G}\rvert \log^2 \lvert\mathsf{G}\rvert)$
③	$\Theta(\lvert\mathsf{G}\rvert^2)$
④	$O(\lvert\mathsf{G}\rvert \cdot 2^{\lvert\mathsf{G}\rvert}),\ \Omega(2^{\lvert\mathsf{G}\rvert^{(1/2-\varepsilon)}})$
⑤	subtyping $\Theta(\lvert\mathsf{T}\rvert^2)$
⑥	$O(U^3 \cdot 2^{3\lvert\Delta\rvert} + U^3 \cdot \log U \cdot 2^{2\lvert\Delta\rvert})$
⑦	$O(\lvert\Delta\rvert \cdot U)$
⑧	$O(\lvert\Delta\rvert \cdot U)$

where $U = \Pi_{i \in I} \mathsf{T}_i$

Figure 2: Projections, typing context properties and complexity from [74, Fig 4 and Tables 1,2,3]

typable endpoint programs and greatly simplifies implementations, this paper has given considerable attention.

Subsequent work [35] has resolved the above concern by introducing a general proof technique for type soundness (subject reduction) of the multiparty session π-calculus. The proof is based on an *association relation* between the behavioural semantics of a global type and that of its endpoint projections. Using this relation, the authors also established session fidelity, deadlock freedom, and liveness for endpoint projection with mergeability.

Nevertheless, the bottom-up strategy offers several distinctive advantages. First, it supports the verification of a broad range of behavioural properties. For example, it can verify communication safety independently of deadlock-freedom or liveness. Secondly, it yields the *maximum* typability with respect to a desired property φ (see below for a formal definition). Thirdly, because local types closely resemble process calculi (e.g., CCS [53] and CSP [31]) or finite-state machines, the bottom-up approach can directly exploit mature off-the-shelf model checkers, such as mCRL2 [70] and SPIN [71], to verify behavioural properties.

The bottom-up approach has been integrated into Scala [1, 68], OCaml [38], Go [60, 47], and Rust [13, 46]. The bottom-up typing system has also been verified by Rocq [29, 17, 18, 30] and Why3 [26].

Limitations of Top-Down Strategy. While [35] solves some concerns in the literature, there is one shortcoming of the top-down strategy–*typability is limited* since it depends on the EPP algorithm being used. If G is not projectable by the chosen EPP algorithm, either G is not *implementable* (there exists no correct set of local types) or this particular EPP algorithm rejects some implementable G. If the EPP is too limited, many safe and live processes become untypable. As expected, computationally cheaper EPP algorithms tend to give fewer typable processes than more expensive ones [74]. See 2.

Limitations of the Bottom-Up Strategy. Since the bottom-up strategy directly verifies behavioural properties of local types, its main drawback is its *computational cost*. Verifying safety, deadlock-freedom and liveness of a collection of local types is PSPACE-complete in the synchronous setting [74] and *undecidable* in the asynchronous setting [5]. By contrast, the top-down strategy requires only endpoint projection and subtyping, both of which can be decided for most of the usecases, polynomial time in the size of the types [74].

The First Summary. The top-down and bottom-up strategies offer complementary trade-offs. The top-down strategy requires an upfront global protocol specification and endpoint projection, but enables computationally efficient verification and provides a human-readable protocol specification. Hence we can guarantee the *protocol conformance*. In contrast, the bottom-up strategy avoids the need for a global specification, but incurs significantly higher verification costs, which are PSPACE-complete in the synchronous setting and undecidable in the asynchronous setting.

2.2 Expressiveness as Typability

Recent work by Pischke and Yoshida [65] has established a surprising result concerning the expressive power (typability) of the top-down and bottom-up typing systems with respect to the *liveness* property under synchronous semantics.

Consider a participant $\mathbf{p_i} :: P_i$, where process P_i implements participant $\mathbf{p_i}$ and has session type $\mathsf{T_i}$. A *multiparty session* is a parallel composition of participants

$$M = \mathbf{p_1} :: P_1 \mid \cdots \mid \mathbf{p_n} :: P_n$$

The bottom-up typing judgement is defined as: $\vdash^{\mathsf{bot}} M : \Delta$ where Δ is a *safe typing context* mapping each participant $\mathbf{p_i}$ to its type $\mathsf{T_i}$. "Safety" on Δ is the minimum requirement needed to guarantee type soundness of M.

Using this judgement, we define the set of typable live sessions $\mathbb{P}_{\mathtt{bot}}$ as:

$$\mathbb{P}_{\mathtt{bot}} = \{ M \mid \vdash^{\mathsf{bot}} M : \Delta \quad \text{such that } \Delta \text{ is live} \}$$

Whenever Δ is live, the session M satisfies liveness [67, 35]. Furthermore, $\mathbb{P}_{\mathtt{bot}}$ is preserved under reduction: if $M \in \mathbb{P}_{\mathtt{bot}}$ and $M \to M'$, then $M' \in \mathbb{P}_{\mathtt{bot}}$. Consequently, $\mathbb{P}_{\mathtt{bot}}$ provides a *complete characterisation of liveness*: a multiparty session is typable and live if and only if it belongs to $\mathbb{P}_{\mathtt{bot}}$.

For the top-down system, each local type $\mathsf{T_i}$ is obtained by projecting a global type G onto participant $\mathbf{p_i}$, written $\mathsf{G} \upharpoonright_{\mathsf{p_i}} \mathsf{T_i}$. We therefore define:

$$\mathbb{P}_{\mathtt{top}} = \{ M \mid \vdash^{\mathsf{top}} M : \Delta \quad \text{such that } \mathsf{G} \upharpoonright_{\mathsf{p_i}} \mathsf{T_i} \quad \Delta = \{\mathbf{p_i}{:}\mathsf{T_i}\}_{i \in I} \text{ for some balanced } \mathsf{G} \}$$

By construction, we have: $\mathbb{P}_{\mathtt{top}} \subseteq \mathbb{P}_{\mathtt{bot}}$.

Depending on the EPP algorithm, the inclusion can be strict. That is, a session may belong to $\mathbb{P}_{\mathtt{bot}}$ but not $\mathbb{P}_{\mathtt{top}}$, because the chosen EPP algorithm fails to project a global type that can implement safe and live processes.

The main result of [65] is that this gap disappears under three conditions:

$$\mathbb{P}_{\mathtt{top}} = \mathbb{P}_{\mathtt{bot}}$$

This overturns the common belief that the top-down strategy is inherently incomplete. More precisely, the top-down system is complete with respect to liveness provided that (i) typing includes the subsumption rule induced by *precise synchronous subtyping* $\leqslant$ [24]; (ii) global types are *balanced*, meaning every participant occurring in a subterm is unavoidable; and (iii) endpoint projection is performed using the most expressive (and computationally most expensive) projection algorithm proposed so far, namely the *coinductive full-merging projection* [74].

The proof of this result relies on a *principal global type inference algorithm*, developed in [65], which synthesises a balanced global type from any safe and live typing context (Figure 1c). The details of the synthesis algorithm and its implementation are found in[65].

3 Further Questions on Expressiveness

The equivalence result in [65] establishes that the top-down and bottom-up typing systems have the same expressive power (typability) only for the synchronous MPST calculus with *separated choice*, *precise synchronous subtyping*, and *the liveness property*. This raises the question of whether similar equivalence results hold under other semantics, language features, subtyping relations, and behavioural properties.

A first future direction is *asynchronous semantics*. Pischke et al. [64] have recently extended the top-down framework with coinductive full-merging projection and precise asynchronous subtyping [25], establishing subject reduction and liveness for the asynchronous MPST-calculus by relying on the corresponding bottom-up results. An important open question is whether the top-down and bottom-up typing systems also coincide with respect to asynchronous liveness.

Our recent LLM-based toolchain [50] enables to judge asynchronous subtyping between a larger set of local types than the existing sound algorithms. The tool consists of constraint generation based on beam search with two levels of monitoring: (1) a lightweight derivative-based feasibility check that prunes invalid subtypes; and (2) a widening-based subtyping checker that validates complete candidates before accepted. Because the architecture is modular, this pipeline is reused for the global type inference from a set of given local types. Can LLMs for asynchronous subtyping *approximately* solve this open problem?

A second direction is to consider behavioural properties beyond liveness. The equivalence in [65] is established only for liveness. Whether an analogous result holds for deadlock-freedom remains open, i.e., whether the top-down typing system can completely characterise the deadlock-free processes accepted by the bottom-up framework (if we change the balanced condition).

A third direction is more expressive process calculi (in the sense of encodability). The current expressiveness result is limited to separated choice. Extending it to mixed-choice MPST appears substantially more difficult. In the asynchronous setting, Stutz proved that deciding whether a global type with mixed choice admits a deadlock-free implementation is undecidable [69]. The same argument might be true for synchronous semantics. Consequently, no decidable endpoint projection algorithm can completely characterise the deadlock-free behaviour of local types. It is therefore unlikely that any decidable projection algorithm can completely capture liveness for mixed-choice MPST even up to the precise mixed choice subtyping [52]. This suggests that practical top-down systems will necessarily rely on projection algorithms that have given up completeness for decidability and efficiency, as proposed in [3].

Acknowledgements. The work is partially supported by EPSRC EP/T006544/2, EP/T014709/2, EP/Y005244/1, EP/V000462/1, EP/X015955/1, EP/Z0005801/1, Horizon EU TaRDIS 101093006 (UKRI No. 10066667), Advanced Research and Invention Agency (ARIA), and a grant from the Simons Foundation.

References

[1] Adam Barwell, Alceste Scalas, Nobuko Yoshida & Fangyi Zhou (2022): *Generalised Multiparty Session Types with Crash-Stop Failures*. In: *33rd International Conference on Concurrency Theory*, *LIPIcs* 243, Dagstuhl, pp. 35:1–35:25, doi:10.4230/LIPIcs.CONCUR.2022.35.

[2] Adam D. Barwell, Ping Hou, Nobuko Yoshida & Fangyi Zhou (2023): *Designing Asynchronous Multiparty Protocols with Crash-Stop Failures*. In Karim Ali & Guido Salvaneschi, editors: *37th European Conference on Object-Oriented Programming (ECOOP 2023)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 263, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 1:1–1:30,

doi:10.4230/LIPIcs.ECOOP.2023.1. Available at https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ECOOP.2023.1.

[3] Laura Bocchi, Raymond Hu, Adriana Laura Voinea & Simon Thompson (2026): *Mixed Choice in Asynchronous Multiparty Session Types*. *Proc. ACM Program. Lang.* 10(OOPSLA1), doi:10.1145/3798256. Available at https://doi.org/10.1145/3798256.

[4] Jelle Bouma, Stijn de Gouw & Sung-Shik Jongmans (2023): *Multiparty Session Typing in Java, Deductively*. In Sriram Sankaranarayanan & Natasha Sharygina, editors: *Tools and Algorithms for the Construction and Analysis of Systems - 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22-27, 2023, Proceedings, Part II*, *Lecture Notes in Computer Science* 13994, Springer, pp. 19–27, doi:10.1007/978-3-031-30820-8_3. Available at https://doi.org/10.1007/978-3-031-30820-8_3.

[5] Daniel Brand & Pitro Zafiropulo (1983): *On Communicating Finite-State Machines*. *Journal of the ACM* 30(2), pp. 323–342, doi:10.1145/322374.322380.

[6] Filipe Casal, Andreia Mordido & Vasco T. Vasconcelos (2022): *Mixed sessions*. *Theoretical Computer Science* 897, pp. 23–48, doi:https://doi.org/10.1016/j.tcs.2021.08.005. Available at https://www.sciencedirect.com/science/article/pii/S0304397521004631.

[7] David Castro-Perez, Francisco Ferreira, Lorenzo Gheri & Nobuko Yoshida (2021): *Zooid: a DSL for certified multiparty computation: from mechanised metatheory to certified multiparty processes*. In Stephen N. Freund & Eran Yahav, editors: *PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021*, ACM, pp. 237–251, doi:10.1145/3453483.3454041. Available at https://doi.org/10.1145/3453483.3454041.

[8] David Castro-Perez, Francisco Ferreira & Sung-Shik Jongmans (2026): *A Synthetic Reconstruction of Multiparty Session Types*. *Proc. ACM Program. Lang.* 10(POPL), doi:10.1145/3776692. Available at https://doi.org/10.1145/3776692.

[9] David Castro-Perez, Raymond Hu, Sung-Shik Jongmans, Nicholas Ng & Nobuko Yoshida (2019): *Distributed programming using role-parametric session types in go: statically-typed endpoint APIs for dynamically-instantiated communication structures*. *Proc. ACM Program. Lang.* 3(POPL), pp. 29:1–29:30, doi:10.1145/3290342.

[10] Minas Charalambides, Peter Dinges & Gul Agha (2012): *Parameterized Concurrent Multi-Party Session Types*. In Natallia Kokash & António Ravara, editors: *Proceedings 11th International Workshop on Foundations of Coordination Languages and Self Adaptation, FOCLASA 2012, Newcastle, U.K., September 8, 2012*, *EPTCS* 91, pp. 16–30, doi:10.4204/EPTCS.91.2. Available at https://doi.org/10.4204/EPTCS.91.2.

[11] Minas Charalambides, Peter Dinges & Gul A. Agha (2016): *Parameterized, concurrent session types for asynchronous multi-actor interactions*. *Sci. Comput. Program.* 115-116, pp. 100–126, doi:10.1016/J.SCICO.2015.10.006. Available at https://doi.org/10.1016/j.scico.2015.10.006.

[12] Guillermina Cledou, Luc Edixhoven, Sung-Shik Jongmans & José Proença (2022): *API Generation for Multiparty Session Types, Revisited and Revised Using Scala 3*. In Karim Ali & Jan Vitek, editors: *36th European Conference on Object-Oriented Programming (ECOOP 2022)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 222, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 27:1–27:28. Available at https://drops.dagstuhl.de/opus/volltexte/2022/16255.

[13] Zak Cutner, Nobuko Yoshida & Martin Vassor (2022): *Deadlock-Free Asynchronous Message Reordering in Rust with Multiparty Session Types*. In: *27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '22* abs/2112.12693, ACM, pp. 261–246, doi:10.1145/3503221.3508404.

[14] Romain Demangeon, Kohei Honda, Raymond Hu, Rumyana Neykova & Nobuko Yoshida (2015): *Practical Interruptible Conversations: Distributed Dynamic Verification with Multiparty Session Types and Python*. *Formal Methods Syst. Des.* 46(3), pp. 197–225, doi:10.1007/s10703-014-0218-8.

[15] Lavinia Egidi, Paola Giannini & Lorenzo Ventura (2022): *Multiparty-session-types Coordination for Core Erlang*. In Hans-Georg Fill, Marten van Sinderen & Leszek A. Maciaszek, editors: *Proceedings of the 17th International Conference on Software Technologies, ICSOFT 2022, Lisbon, Portugal, July 11-13, 2022*, SCITEPRESS, pp. 532–541, doi:10.5220/0011316600003266. Available at https://doi.org/10.5220/0011316600003266.

[16] Burak Ekici, Tadayoshi Kamegai & Nobuko Yoshida (2025): *Formalising Subject Reduction and Progress for Multiparty Session Processes*. In Yannick Forster & Chantal Keller, editors: *16th International Conference on Interactive Theorem Proving, ITP 2025, September 28 to October 1, 2025, Reykjavik, Iceland*, *LIPIcs* 352, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 19:1–19:23, doi:10.4230/LIPICS.ITP.2025.19. Available at https://doi.org/10.4230/LIPIcs.ITP.2025.19.

[17] Burak Ekici & Nobuko Yoshida (2024): *Completeness of Asynchronous Session Tree Subtyping in Coq*. In Yves Bertot, Temur Kutsia & Michael Norrish, editors: *15th International Conference on Interactive Theorem Proving, ITP 2024, September 9-14, 2024, Tbilisi, Georgia*, *LIPIcs* 309, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 13:1–13:20, doi:10.4230/LIPICS.ITP.2024.13. Available at https://doi.org/10.4230/LIPIcs.ITP.2024.13.

[18] Burak Ekici & Nobuko Yoshida (2026): *Formalising Asynchronous Session Subtyping*. *ACM Trans. Comput. Logic* 27(3), doi:10.1145/3815176. Available at https://doi.org/10.1145/3815176.

[19] Francisco Ferreira & Sung-Shik Jongmans (2023): *Oven: Safe and Live Communication Protocols in Scala, using Synthetic Behavioural Type Analysis*. In René Just & Gordon Fraser, editors: *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, ACM, pp. 1511–1514, doi:10.1145/3597926.3604926. Available at https://doi.org/10.1145/3597926.3604926.

[20] Simon Fowler (2016): *An Erlang Implementation of Multiparty Session Actors*. In Massimo Bartoletti, Ludovic Henrio, Sophia Knight & Hugo Torres Vieira, editors: *Proceedings 9th Interaction and Concurrency Experience, ICE 2016, Heraklion, Greece, 8-9 June 2016*, *EPTCS* 223, pp. 36–50, doi:10.4204/EPTCS.223.3.

[21] Simon Fowler & Raymond Hu (2026): *Spark Now: Safe Actor Programming with Multiparty Session Types*. Available at https://simonjf.com/writing/eventactors.pdf. To appear in OOPSLA'26.

[22] Simon Fowler & Raymond Hu (2026): *Speak Now: Safe Actor Programming with Multiparty Session Types*. *Proc. ACM Program. Lang.* 10(OOPSLA1), doi:10.1145/3798267. Available at https://doi.org/10.1145/3798267.

[23] Lorenzo Gheri, Ivan Lanese, Neil Sayers, Emilio Tuosto & Nobuko Yoshida (2022): *Design-By-Contract for Flexible Multiparty Session Protocols*. In Karim Ali & Jan Vitek, editors: *36th European Conference on Object-Oriented Programming, ECOOP 2022, June 6-10, 2022, Berlin, Germany*, *LIPIcs* 222, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 8:1–8:28, doi:10.4230/LIPICS.ECOOP.2022.8. Available at https://doi.org/10.4230/LIPIcs.ECOOP.2022.8.

[24] Silvia Ghilezan, Svetlana Jakšić, Jovanka Pantović, Alceste Scalas & Nobuko Yoshida (2019): *Precise subtyping for synchronous multiparty sessions*. *Journal of Logical and Algebraic Methods in Programming* 104, pp. 127–173, doi:https://doi.org/10.1016/j.jlamp.2018.12.002. Available at https://www.sciencedirect.com/science/article/pii/S2352220817302237.

[25] Silvia Ghilezan, Jovanka Pantović, Ivan Prokić, Alceste Scalas & Nobuko Yoshida (2023): *Precise Subtyping for Asynchronous Multiparty Sessions*. *ACM Trans. Comput. Logic* 24(2), doi:10.1145/3568422. Available at https://doi.org/10.1145/3568422.

[26] Marco Giunti & Nobuko Yoshida (2025): *Iso-Recursive Multiparty Sessions and their Automated Verification*. In Viktor Vafeiadis, editor: *Programming Languages and Systems - 34th European Symposium on Programming, ESOP 2025, Held as Part of the International Joint Conferences on Theory and Practice of Software, ETAPS 2025, Hamilton, ON, Canada, May 3-8, 2025, Proceedings, Part I*, *Lecture Notes in Computer Science* 15694, Springer, pp. 349–378, doi:10.1007/978-3-031-91118-7_14. Available at https://doi.org/10.1007/978-3-031-91118-7_14.

[27] Daniele Gorla (2010): *Towards a unified approach to encodability and separation results for process calculi*. *Inf. Comput.* 208(9), pp. 1031–1053, doi:10.1016/j.ic.2010.05.002. Available at `http://dx.doi.org/10.1016/j.ic.2010.05.002`.

[28] Paul Harvey, Simon Fowler, Ornela Dardha & Simon J. Gay (2021): *Multiparty Session Types for Safe Runtime Adaptation in an Actor Language*. In Anders Møller & Manu Sridharan, editors: *35th European Conference on Object-Oriented Programming, ECOOP 2021, July 11-17, 2021, Aarhus, Denmark (Virtual Conference)*, *LIPIcs* 194, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 10:1–10:30, doi:10.4230/LIPICS.ECOOP.2021.10. Available at `https://doi.org/10.4230/LIPIcs.ECOOP.2021.10`.

[29] Jonas Kastberg Hinrichsen, Jules Jacobs & Robbert Krebbers (2024): *Multris: Functional Verification of Multiparty Message Passing in Separation Logic*. *Proc. ACM Program. Lang.* 8(OOPSLA2), pp. 1446–1474, doi:10.1145/3689762. Available at `https://doi.org/10.1145/3689762`.

[30] Jonas Kastberg Hinrichsen, Iwan Quémerais & Lars Birkedal (2026): *Mixtris: Mechanised Higher-Order Separation Logic for Mixed Choice Multiparty Message Passing*. *Proc. ACM Program. Lang.* 10(OOPSLA1), doi:10.1145/3798224. Available at `https://doi.org/10.1145/3798224`.

[31] C. A. R. Hoare (1985): *Communicating Sequential Processes*. Prentice-Hall, Inc.

[32] Kohei Honda, Nobuko Yoshida & Marco Carbone (2008): *Multiparty Asynchronous Session Types*. In: *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '08, Association for Computing Machinery, New York, NY, USA, pp. 273–284, doi:10.1145/1328438.1328472.

[33] Kohei Honda, Nobuko Yoshida & Marco Carbone (2016): *Multiparty Asynchronous Session Types*. *Journal of the ACM* 63(1), pp. 9:1–9:67, doi:10.1145/2827695.

[34] Ping Hou, Nicolas Lagaillardie & Nobuko Yoshida (2024): *Fearless Asynchronous Communications with Timed Multiparty Session Protocols*. In Jonathan Aldrich & Guido Salvaneschi, editors: *38th European Conference on Object-Oriented Programming, ECOOP 2024, September 16-20, 2024, Vienna, Austria*, *LIPIcs* 313, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 19:1–19:30, doi:10.4230/LIPICS.ECOOP.2024.19. Available at `https://doi.org/10.4230/LIPIcs.ECOOP.2024.19`.

[35] Ping Hou, Nobuko Yoshida & Iona Kuhn (2026): *Less is more revisited: Association with global protocols and multiparty sessions*. *Theoretical Computer Science* 1076, p. 115873, doi:https://doi.org/10.1016/j.tcs.2026.115873. Available at `https://www.sciencedirect.com/science/article/pii/S0304397526001325`.

[36] Raymond Hu & Nobuko Yoshida (2016): *Hybrid Session Verification through Endpoint API Generation*. In: *19th International Conference on Fundamental Approaches to Software Engineering*, *LNCS* 9633, Springer, Berlin, Heidelberg, pp. 401–418, doi:10.1007/978-3-662-49665-7_24.

[37] Raymond Hu & Nobuko Yoshida (2017): *Explicit Connection Actions in Multiparty Session Types*. In: *FASE*, *LNCS* 10202, Springer, pp. 116–133, doi:10.1007/978-3-662-54494-5_7.

[38] Keigo Imai, Julien Lange & Rumyana Neykova (2022): *Kmclib: Automated Inference and Verification of Session Types from OCaml Programs*. In Dana Fisman & Grigore Rosu, editors: *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*, *Lecture Notes in Computer Science* 13243, Springer, pp. 379–386, doi:10.1007/978-3-030-99524-9_20. Available at `https://doi.org/10.1007/978-3-030-99524-9_20`.

[39] Keigo Imai, Rumyana Neykova, Nobuko Yoshida & Shoji Yuen (2020): *Multiparty Session Programming with Global Protocol Combinators*. In: *34th European Conference on Object-Oriented Programming*, *LIPIcs* 166, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, pp. 9:1–9:30, doi:10.4230/LIPIcs.ECOOP.2020.9.

[40] Jules Jacobs, Stephanie Balzer & Robbert Krebbers (2022): *Multiparty GV: functional multiparty session types with certified deadlock freedom*. Proc. ACM Program. Lang. 6(ICFP), pp. 466–495, doi:10.1145/3547638. Available at https://doi.org/10.1145/3547638.

[41] Sung-Shik Jongmans (2025): *Multiparty Session Typing, Embedded*. In Arie Gurfinkel & Marijn Heule, editors: *Tools and Algorithms for the Construction and Analysis of Systems - 31st International Conference, TACAS 2025, Held as Part of the International Joint Conferences on Theory and Practice of Software, ETAPS 2025, Hamilton, ON, Canada, May 3-8, 2025, Proceedings, Part I*, *Lecture Notes in Computer Science* 15696, Springer, pp. 145–164, doi:10.1007/978-3-031-90643-5_8. Available at https://doi.org/10.1007/978-3-031-90643-5_8.

[42] Omer Keskin, Nobuko Yoshida & Rob van Glabbeek (2026): *Formally Verified Liveness with Multiparty Session Types in Rocq*. In: *17th International Conference on Interactive Theorem Proving, 2026, Lisbon, Portugal*, *LIPIcs* 382, Schloss Dagstuhl - Leibniz-Zentrum f"ur Informatik, pp. 4:1–4:21, doi:https://doi.org/10.4230/LIPIcs.ITP.2026.4.

[43] Shunsuke Kimura & Keigo Imai (2020): *Fluent Session Programming in C#*. In Stephanie Balzer & Luca Padovani, editors: *Proceedings of the 12th International Workshop on Programming Language Approaches to Concurrency- and Communication-cEntric Software, PLACES'2020, Dublin, Ireland, 26th April 2020*, *EPTCS* 314, pp. 61–75, doi:10.4204/EPTCS.314.6. Available at https://doi.org/10.4204/EPTCS.314.6.

[44] Jonathan King, Nicholas Ng & Nobuko Yoshida (2019): *Multiparty Session Type-safe Web Development with Static Linearity*. In Francisco Martins & Dominic Orchard, editors: *Proceedings Programming Language Approaches to Concurrency- and Communication-cEntric Software, PLACES@ETAPS 2019, Prague, Czech Republic, 7th April 2019*, *EPTCS* 291, pp. 35–46, doi:10.4204/EPTCS.291.4. Available at https://doi.org/10.4204/EPTCS.291.4.

[45] Dimitrios Kouzapas, Ornela Dardha, Roly Perera & Simon J. Gay (2018): *Typechecking protocols with Mungo and StMungo: A session type toolchain for Java*. Sci. Comput. Program. 155, pp. 52–75, doi:10.1016/J.SCICO.2017.10.006. Available at https://doi.org/10.1016/j.scico.2017.10.006.

[46] Nicolas Lagaillardie, Rumyana Neykova & Nobuko Yoshida (2022): *Stay Safe Under Panic: Affine Rust Programming with Multiparty Session Types*. In Karim Ali & Jan Vitek, editors: *36th European Conference on Object-Oriented Programming, ECOOP 2022, June 6-10, 2022, Berlin, Germany*, *LIPIcs* 222, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 4:1–4:29, doi:10.4230/LIPICS.ECOOP.2022.4. Available at https://doi.org/10.4230/LIPIcs.ECOOP.2022.4.

[47] Julien Lange, Nicholas Ng, Bernardo Toninho & Nobuko Yoshida (2018): *A Static Verification Framework for Message Passing in Go using Behavioural Types*. In: *40th International Conference on Software Engineering*, ACM, pp. 1137–1148, doi:10.1145/3180155.3180157.

[48] Jens Kanstrup Larsen, Alceste Scalas, Guy Amir, Jules Jacobs, Jana Wagemaker & Nate Foster (2026): *NEST: Network Enforced Session Types*. In Robbert Krebbers & Alexandra Silva, editors: *40th European Conference on Object-Oriented Programming (ECOOP 2026)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 372, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 17:1–17:31, doi:10.4230/LIPIcs.ECOOP.2026.17. Available at https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ECOOP.2026.17.

[49] Elaine Li & Thomas Wies (2025): *Certified Implementability of Global Multiparty Protocols*. In Yannick Forster & Chantal Keller, editors: *16th International Conference on Interactive Theorem Proving, ITP 2025, September 28 to October 1, 2025, Reykjavik, Iceland*, *LIPIcs* 352, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 15:1–15:20, doi:10.4230/LIPICS.ITP.2025.15. Available at https://doi.org/10.4230/LIPIcs.ITP.2025.15.

[50] Yang Li, Ping Hou & Nobuko Yoshida (2026): *Specification-Guided Synthesis of Deadlock-Free Communication Protocol Refinements with Large Language Models*. In: *The 41st IEEE/ACM International Conference on Automated Software Engineering, Munich, Germany*, pp. –.

[51] Hugo A. López, Eduardo R. B. Marques, Francisco Martins, Nicholas Ng, César Santos, Vasco Thudichum Vasconcelos & Nobuko Yoshida (2015): *Protocol-Based Verification of Message-Passing Parallel Programs*. In: *2015 ACM International Conference on Object Oriented Programming Systems Languages and Applications / SPLASH '15*, ACM, pp. 280–298, doi:10.1145/2814270.2814302.

[52] Jake Masters & Nobuko Yoshida (2026): *Mixed Choice Multiparty Session Types, Precisely*. Available at `https://arxiv.org/abs/2608.10704`. To appear in ESOP'27.

[53] Robin Milner (1989): *Communication and Concurrency*. Prentics Hall.

[54] Anson Miu, Francisco Ferreira, Nobuko Yoshida & Fangyi Zhou (2021): *Communication-Safe Web Programming in TypeScript with Routed Multiparty Session Types*. In: *International Conference on Compiler Construction*, CC, pp. 94–106, doi:10.1145/3446804.3446854.

[55] Rumyana Neykova, Raymond Hu, Nobuko Yoshida & Fahd Abdeljallal (2018): *A Session Type Provider: Compile-time API Generation for Distributed Protocols with Interaction Refinements in F#*. In: *27th International Conference on Compiler Construction*, ACM, pp. 128–138, doi:10.1145/3178372.3179495.

[56] Rumyana Neykova & Nobuko Yoshida (2017): *Let It Recover: Multiparty Protocol-Induced Recovery*. In: *CC*, ACM, pp. 98–108, doi:10.1145/3033019.3033031.

[57] Rumyana Neykova & Nobuko Yoshida (2017): *Multiparty Session Actors*. *Logical Methods in Computer Science* 13, pp. 1–30, doi:10.23638/LMCS-13(1:17)2017.

[58] Rumyana Neykova, Nobuko Yoshida & Raymond Hu (2013): *SPY: Local Verification of Global Protocols*. In Axel Legay & Saddek Bensalem, editors: *Runtime Verification - 4th International Conference, RV 2013, Rennes, France, September 24-27, 2013. Proceedings*, *Lecture Notes in Computer Science* 8174, Springer, pp. 358–363, doi:10.1007/978-3-642-40787-1_25. Available at `https://doi.org/10.1007/978-3-642-40787-1_25`.

[59] Nicholas Ng, Jose Gabriel de Figueiredo Coutinho & Nobuko Yoshida (2015): *Protocols by Default*. In Björn Franke, editor: *Compiler Construction*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 212–232.

[60] Nicholas Ng & Nobuko Yoshida (2016): *Static Deadlock Detection for Concurrent Go by Global Session Graph Synthesis*. In: *25th International Conference on Compiler Construction*, ACM, pp. 174–184, doi:10.1145/2892208.2892232.

[61] Nicholas Ng, Nobuko Yoshida & Kohei Honda (2012): *Multiparty Session C: Safe Parallel Programming with Message Optimisation*. In: *50th International Conference on Objects, Models, Components, Patterns*, *LNCS* 7304, Springer, pp. 202–218, doi:10.1007/978-3-642-30561-0_15.

[62] Nicholas Ng, Nobuko Yoshida, Xinyu Niu, Kuen Hung Tsoi & Wayne Luk (2012): *Session Types: Towards safe and fast reconfigurable programming*. *ACM SIGARCH Computer Architecture News* 40, pp. 22–27, doi:10.1145/2460216.2460221.

[63] Kirstin Peters & Nobuko Yoshida (2024): *Separation and Encodability in Mixed Choice Multiparty Sessions*. In Pawel Sobocinski, Ugo Dal Lago & Javier Esparza, editors: *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2024, Tallinn, Estonia, July 8-11, 2024*, ACM, pp. 62:1–62:15, doi:10.1145/3661814.3662085. Available at `https://doi.org/10.1145/3661814.3662085`.

[64] Kai Pischke, Jake Masters & Nobuko Yoshida (2026): *Asynchronous Global Protocols, Precisely*. *Information and Computation*, p. 105501, doi:https://doi.org/10.1016/j.ic.2026.105501. Available at `https://www.sciencedirect.com/science/article/pii/S0890540126000982`.

[65] Kai Pischke & Nobuko Yoshida (2026): *Top-down = Bottom-up: Sound and Complete Characterisations of Liveness by Multiparty Global Protocols*. *Proc. ACM Program. Lang.* 10(OOPSLA2), doi:10.1145/3839538.

[66] Alceste Scalas, Ornela Dardha, Raymond Hu & Nobuko Yoshida (2017): *A Linear Decomposition of Multiparty Sessions for Safe Distributed Programming*. In: *ECOOP*, *LIPIcs* 74, Schloss Dagstuhl, pp. 24:1–24:31, doi:10.4230/LIPIcs.ECOOP.2017.24.

[67] Alceste Scalas & Nobuko Yoshida (2019): *Less Is More: Multiparty Session Types Revisited*. *Proceedings of the ACM on Programming Languages* 3(POPL), pp. 1–29, doi:10.1145/3290343.

[68] Alceste Scalas, Nobuko Yoshida & Elias Benussi (2019): *Verifying message-passing programs with dependent behavioural types*. In: *Programming Language Design and Implementation*, ACM, pp. 502–516, doi:10.1145/3314221.3322484.

[69] Felix Stutz (2023): *The Power of Choice: Implementability of Asynchronous Communication Protocols*. Dr. rer. nat. thesis, University of Kaiserslautern–Landau, Kaiserslautern, Germany. Available at `https://kluedo.ub.rptu.de/files/8077/Doctoral-Thesis-Felix-Stutz.pdf`. Reviewer: Prof. Dr. Rupak Majumdar; Defense date: 15 December 2023.

[70] mCRL2 team (2025): *mCRL2 homepage*. `https://mcrl2.org/web/index.html`.

[71] SPIN team (2025): *SPIN homepage*. `https://spinroot.com/spin/whatispin.html`.

[72] Dawit Legesse Tirore, Jesper Bengtson & Marco Carbone (2025): *Multiparty Asynchronous Session Types: A Mechanised Proof of Subject Reduction*. In Jonathan Aldrich & Alexandra Silva, editors: *39th European Conference on Object-Oriented Programming, ECOOP 2025, June 30 to July 2, 2025, Bergen, Norway*, *LIPIcs* 333, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 31:1–31:30, doi:10.4230/LIPICS.ECOOP.2025.31. Available at `https://doi.org/10.4230/LIPIcs.ECOOP.2025.31`.

[73] Dawit Legesse Tirore, Jesper Bengtson & Marco Carbone (2025): *A Sound and Complete Projection for Global Types*. *J. Autom. Reason.* 69(2), p. 14, doi:10.1007/S10817-025-09726-9. Available at `https://doi.org/10.1007/s10817-025-09726-9`.

[74] Thien Udomsrirungruang & Nobuko Yoshida (2025): *Top-Down or Bottom-Up? Complexity Analyses of Synchronous Multiparty Session Types*. *Proc. ACM Program. Lang.* 9(POPL), pp. 1040–1071, doi:10.1145/3704872. Available at `https://doi.org/10.1145/3704872`.

[75] Martin Vassor & Nobuko Yoshida (2024): *Refinements for Multiparty Message-Passing Protocols: Specification-Agnostic Theory and Implementation*. In Jonathan Aldrich & Guido Salvaneschi, editors: *38th European Conference on Object-Oriented Programming, ECOOP 2024, September 16-20, 2024, Vienna, Austria*, *LIPIcs* 313, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 41:1–41:29, doi:10.4230/LIPICS.ECOOP.2024.41. Available at `https://doi.org/10.4230/LIPIcs.ECOOP.2024.41`.

[76] Malte Viering, Raymond Hu, Patrick Eugster & Lukasz Ziarek (2021): *A Multiparty Session Typing Discipline for Fault-Tolerant Event-Driven Distributed Programming*. *Proc. ACM Program. Lang.* 5(OOPSLA), doi:10.1145/3485501. Available at `https://doi.org/10.1145/3485501`.

[77] Nobuko Yoshida (2024): *Programming Language Implementations with Multiparty Session Types*. In: *Active Object Languages: Current Research Trends*, Springer Nature Switzerland, pp. 147–165, doi:10.1007/978-3-031-51060-1_6.

[78] Nobuko Yoshida, Fangyi Zhou & Francisco Ferreira (2021): *Communicating Finite State Machines and an Extensible Toolchain for Multiparty Session Types*. In: *23rd International Symposium on Fundamentals of Computation Theory*, *LNCS* 12867, Springer, pp. 18–35, doi:10.1007/978-3-030-86593-1_2.

[79] Fangyi Zhou, Francisco Ferreira, Raymond Hu, Rumyana Neykova & Nobuko Yoshida (2020): *Statically Verified Refinements for Multiparty Protocols*. In: *OOPSLA 2020: Conference on Object-Oriented Programming Systems, Languages and Applications*, *PACMPL* 4, ACM, pp. 148:1–148:30, doi:10.1145/3428216.