# On the Preciseness of Subtyping in Session Types: 10 Years Later

Tzu-Chun Chen
Microsoft, Germany
gina.chen@microsoft.com

Mariangiola Dezani-Ciancaglini
University of Torino, Italy
dezani@di.unito.it

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

## ABSTRACT

The PPDP Most Influential Paper 10-Year Award for our work [11] was a delightful surprise. We subsequently reviewed the subsequent literature to see how our results have been utilised. This short note aims to capture crucial references without missing too many.

## CCS CONCEPTS

- **Theory of computation** → **Process calculi**; **Type theory**.

## KEYWORDS

$\pi$-calculus, Session types, Subtyping, Soundness and Completeness

**ACM Reference Format:**

Tzu-Chun Chen, Mariangiola Dezani-Ciancaglini, and Nobuko Yoshida. 2024. On the Preciseness of Subtyping in Session Types: 10 Years Later. In *Proceedings of the 26th Symposium on Principles and Practice of Declarative Programming, PPDP 2024, Milano, Italy, September 10-11, 2024.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/NNNNNNN.NNNNNNN

*Session types* [23, 24, 26, 27, 36] are a successful formalism to structure interaction and to reason over communicating processes and their behaviour. The basic idea is to introduce a new form of polymorphism which allows the typing of channel names by structured sequences of types, abstractly representing the traces of channel usages. A crucial choice is whether the processes communicate synchronously or asynchronously.

*Subtyping* enhances the expressiveness of session types. For synchronous processes, two distinct subtyping approaches have been proposed: one that allows the safe substitution of channels [18] and another that allows the safe substitution of processes [13]. In our work [11], focused on replacing processes, we adopted the latter approach. We dub this subtyping *synchronous subtyping*. This subtyping has been extended to accommodate asynchronous communications [32], essentially capturing the freedom of outputs typical in such settings.

*Preciseness* of subtyping was first defined for the call-by-value $\lambda$-calculus with iso-recursive types in [30]. A subtyping is precise if it is both sound and complete. A subtyping relation is *sound* if no typable program is incorrect. It is *complete* if there is no strictly larger sound subtyping relation.

Our first major result in [11] was demonstrating the preciseness of synchronous subtyping for synchronous sessions. A key element in proving completeness was the construction of processes that characterise types.

Asynchronous communication is modelled using queues: output processes put messages in queues, while input processes read messages from these queues [26]. In such a scenario, the types should ensure not only deadlock-freedom, i.e., that input processes will always find messages, but also *orphan message-freedom*, i.e., that all messages in queues will eventually be consumed. This notion of soundness was first formulated in [11] and then widely adopted in the literature. The synchronous subtyping that is sound for synchronous sessions is also sound for asynchronous sessions, but it is not complete for asynchronous sessions. The subtyping proposed in [32] enjoys subject reduction, but it is unsound for asynchronous sessions because it does not ensure orphan message-freedom. It is also incomplete for asynchronous sessions. These incompleteness and unsoundness results are demonstrated in [11] through examples. The main achievement of [11] is the definition of a new subtyping (dubbed *asynchronous subtyping*) together with the proof of its preciseness for asynchronous sessions. Again, a key aspect of proving completeness is the construction of processes that characterise types, even though this construction is much more complex than in the case of synchronous subtyping.

Preciseness can be defined operationally by means of processes or denotationally using type interpretations. Both forms of preciseness are proved for the synchronous and the asynchronous subtyping in [11]. Notably, [11] was the first paper discussing preciseness in the context of process calculi.

The most direct follow-up of [11] are papers discussing various aspects of preciseness. In [15], denotational and operational preciseness of subtyping for some $\lambda$-calculi and mobile processes are compared. While in [11] only binary sessions are considered, in [16] the operational and denotational preciseness of the synchronous subtyping for synchronous *multiparty sessions types* (*MPST*) is proved. The novelty of this paper is the introduction of characteristic global types to show the operational completeness.

In [10], new results about the uniqueness of precise subtyping relations are provided. In the same paper the approach of [11] is generalised to session initialisation and communication of expressions including shared channels. In [19] the preciseness of the synchronous subtyping for synchronous MPST is proved using a novel coinductive treatment of global type projections, based on global and local type trees.

Certainly, the most interesting development in this line of research is contained in the papers [20, 21], where the first formalisation of the precise subtyping relation for asynchronous MPST is presented. The proof is based on a novel session decomposition technique, from full session types (including internal/external choices) into single

---

*PPDP '24, September 10-11, 2024, Milano, Italy*

ACM ISBN 9-8-4007-0969-297…$15.00
https://doi.org/10.1145/NNNNNNN.NNNNNNN

input/output session trees (without choices). This session decomposition technique expresses the subtyping relation as a composition of refinement relations between single input/output trees and provides a simple reasoning principle for optimising the order between asynchronous messages. A related denotational semantics of action permutations under different multiparty communication queues and buffers is studied in [14]. There the permutation defined in [21] is modelled as a valid semantic transformation which does not cause deadlock.

A second line of research inspired by [11] addresses the undecidability of asynchronous subtyping. In [4] a core undecidable subtyping relation (obtained by imposing limitations on the structure of types) is devised. As a result of this initial undecidability finding, the asynchronous subtyping and the subtypings of [31, 32] are all shown to be undecidable. The undecidability proof for asynchronous subtyping in [29] relies on a new Turing-complete subclass of two-party communicating finite-state machines, demonstrating that asynchronous subtyping is equivalent to the halting problem for this class of machines. The undecidability result for asynchronous session subtyping is used to obtain an undecidability result for asynchronous contract refinement in [7] and for asynchronous communicating finite state machines in [8]. A novel variant of session subtyping that leverages the notion of controllability from service contract theory and that is a sound characterisation of fair refinement is proposed in [6]. Also, this subtyping and the fair refinement are undecidable.

A natural reaction to the undecidability of asynchronous subtyping is the search for either decidable restrictions or algorithms that can terminate without providing a definitive answer. The decidability of a fragment that does not impose any limitation on communication buffers and allows both the subtype and the supertype to include multiple choices is shown in [5]. The algorithm in [3] is based on a tree representation of the coinductive definition of asynchronous subtyping; this tree could be infinite, and the algorithm checks for the presence of finite witnesses of infinite successful subtrees. The proposal of [2] uses sets of traces instead of trees. The obtained algorithm applies abstract interpretation techniques.

The promotion of asynchronous subtyping incorporation in applications is also an interesting follow-up of [11]. The first work which informally introduces the idea of asynchronous subtyping in practice is [25], where asynchronous multiparty subtyping enables the programmer to permute the order of messages for performance gain without introducing deadlock. The asynchronous subtyping is used to model the double buffering protocol [28]. This approach was implemented and evaluated in C [35, 38] and MPI-C [33, 34] in the context of high-performance computing.

The tool presented in [1] integrates several algorithms for checking subtyping that can be invoked from an easy-to-use Python GUI. This interface allows users to input, using standard session type syntax, two types: the candidate subtype and supertype.

The recent work [9] proposes CAMP, which is a static performance analysis framework for message-passing concurrent and distributed systems based on MPST. CAMP augments MPST with annotations of communication latency and local computation cost, defined as estimated execution times, which is used to extract cost equations from protocol descriptions and to statically predict the communication cost. CAMP is also extended to analyse asynchronous communication optimised programs. The tool based on cost theory is applicable to different existing benchmarks and use cases in the literature with a wide range of communication protocols, including the implementations in [34, 35].

The Rust programming framework, Rumpsteak [12], incorporates multiparty asynchronous subtyping [21] to optimise asynchronous message-passing in the Rust programming language. Specifically, the authors propose an algorithm for asynchronous subtyping based on the session decomposition technique in [20, 21] that is bounded by a number of iterations and proved to be sound and decidable. They evaluate the performance and expressiveness of Rumpsteak against three previous Rust implementations. Rumpsteak is more efficient and can safely express many more examples by offering arbitrary ordering of messages. The authors also analyse the complexity of the new algorithm and benchmark it against the binary session subtyping algorithm in [3]. The algorithm in [3] turns out to be exponentially slower than Rumpsteak.

Hinrichsen's PhD thesis [22] introduces Actris, a Coq tool that integrates separation logics and asynchronous binary session types with the asynchronous subtyping in [11].

The first formalisation of multiparty asynchronous subtyping within the Coq proof assistant is given in [17]. Session types are decomposed into session trees that do not involve choices, and then a coinductive refinement relation is established over them to govern subtyping. This approach allows for the proof of example subtyping schemas that appear in the literature. Notably, to the best of our knowledge, no other decidable sound algorithm is able to verify all these examples.

We conclude by observing that we were wrong in [11], as we wrote: “Algorithms for checking the synchronous and asynchronous subtypings of the present paper can be easily designed". In fact, while there are algorithms for synchronous subtyping (see [37] and the references there), the asynchronous subtyping is undecidable as discussed above. The challenge of asynchronous subtyping remains intriguingly complex and theoretically rich. This underscores the evolving nature of the field and opens avenues for future exploration.

## ACKNOWLEDGMENTS

We thank Alceste Scalas for his collaboration in writing the journal version [10] of [11]. We are grateful to Ilaria Castellani for her valuable suggestions that improved the presentation of this abstract. The last author is partially supported by EPSRC EP/T006544/2, EP/K011715/1, EP/K034413/1, EP/L00058X/1, EP/N027833/2, EP/N028201/1, EP/T014709/2, EP/V000462/1, EP/X015955/1 and Horizon EU TaRDIS 101093006.

## REFERENCES

[1] Lorenzo Bacchiani, Mario Bravetti, Julien Lange, and Gianluigi Zavattaro. 2021. A Session Subtyping Tool. In *COORDINATION (LNCS, Vol. 12717)*, Ferruccio Damiani and Ornela Dardha (Eds.). Springer, Heidelberg, 90–105. https://doi.org/10.1007/978-3-030-78142-2_6

[2] Laura Bocchi, Andy King, and Maurizio Murgia. 2024. Asynchronous Subtyping by Trace Relaxation. In *TACAS (LNCS, Vol. 14570)*, Bernd Finkbeiner and Laura Kovács (Eds.). Springer, Heidelberg, 207–226. https://doi.org/10.1007/978-3-031-57246-3_12

[3] Mario Bravetti, Marco Carbone, Julien Lange, Nobuko Yoshida, and Gianluigi Zavattaro. 2021. A Sound Algorithm for Asynchronous Session Subtyping and its Implementation. *Log. Methods Comput. Sci.* 17, 1 (2021), 20:1–20:35. https:

//lmcs.episciences.org/7238

[4] Mario Bravetti, Marco Carbone, and Gianluigi Zavattaro. 2017. Undecidability of Asynchronous Session Subtyping. *Inf. Comput.* 256 (2017), 300–320. https://doi.org/10.1016/J.IC.2017.07.010

[5] Mario Bravetti, Marco Carbone, and Gianluigi Zavattaro. 2018. On the Boundary between Decidability and Undecidability of Asynchronous Session Subtyping. *Theor. Comput. Sci.* 722 (2018), 19–51. https://doi.org/10.1016/J.TCS.2018.02.010

[6] Mario Bravetti, Julien Lange, and Gianluigi Zavattaro. 2021. Fair Refinement for Asynchronous Session Types. In *FOSSACS (LNCS, Vol. 12650)*, Stefan Kiefer and Christine Tasson (Eds.). Springer, Heidelberg, 144–163. https://doi.org/10.1007/978-3-030-71995-1_8

[7] Mario Bravetti and Gianluigi Zavattaro. 2019. Relating Session Types and Behavioural Contracts: The Asynchronous Case. In *SEFM (LNCS, Vol. 11724)*, Peter Csaba Ölveczky and Gwen Salaün (Eds.). Springer, Heidelberg, 29–47. https://doi.org/10.1007/978-3-030-30446-1_2

[8] Mario Bravetti and Gianluigi Zavattaro. 2021. Asynchronous Session Subtyping as Communicating Automata Refinement. *Softw. Syst. Model.* 20, 2 (2021), 311–333. https://doi.org/10.1007/S10270-020-00838-X

[9] David Castro-Perez and Nobuko Yoshida. 2020. CAMP: Cost-Aware Multiparty Session Protocol. In *OOPSLA*, Vol. 4. ACM, New York, 155:1–155:30. Issue OOPSLA. https://doi.org/10.1145/3428223

[10] Tzu-Chun Chen, Mariangiola Dezani-Ciancaglini, Alceste Scalas, and Nobuko Yoshida. 2017. On the Preciseness of Subtyping in Session Types. *Log. Methods Comput. Sci.* 13, 2 (2017), 12:1–12:62. https://doi.org/10.23638/LMCS-13(2:12)2017

[11] Tzu-Chun Chen, Mariangiola Dezani-Ciancaglini, and Nobuko Yoshida. 2014. On the Preciseness of Subtyping in Session Types. In *PPDP*, Olaf Chitil, Andy King, and Olivier Danvy (Eds.). ACM Press, New York, 135–146. https://doi.org/10.1145/2643135.2643138

[12] Zak Cutner, Nobuko Yoshida, and Martin Vassor. 2022. Deadlock-Free Asynchronous Message Reordering in Rust with Multiparty Session Types. In *PPoPP*, Vol. abs/2112.12693. ACM, New York, 261–246. https://doi.org/10.1145/3503221.3508404

[13] Romain Demangeon and Kohei Honda. 2012. Nested Protocols in Session Types. In *CONCUR (LNCS, Vol. 7454)*, Maciej Koutny and Irek Ulidowski (Eds.). Springer, Heidelberg, 272–286. https://doi.org/10.1007/978-3-642-32940-1_20

[14] Romain Demangeon and Nobuko Yoshida. 2015. On the Expressiveness of Multiparty Session Types. In *FSTTCS (LIPIcs, Vol. 45)*, Prahladh Harsha and G. Ramalingam (Eds.). Dagstuhl Publishing, Dagstuhl, 560–574. https://doi.org/10.4230/LIPIcs.FSTTCS.2015.560

[15] Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, Svetlana Jaksic, Jovanka Pantovic, and Nobuko Yoshida. 2015. Precise Subtyping for Synchronous Multiparty Sessions. In *PLACES (EPTCS, Vol. 203)*, Simon Gay and Jade Alglave (Eds.). Open Publishing Association, Waterloo, 29–43. https://doi.org/10.4204/EPTCS.203.3

[16] Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, Svetlana Jaksic, Jovanka Pantovic, and Nobuko Yoshida. 2016. Denotational and Operational Preciseness of Subtyping: A Roadmap. In *Theory and Practice of Formal Methods (LNCS, Vol. 9660)*, Erika Ábrahám, Marcello M. Bonsangue, and Einar Broch Johnsen (Eds.). Springer, Heidelberg, 155–172. https://doi.org/10.1007/978-3-319-30734-3_12

[17] Burak Ekici and Nobuko Yoshida. 2024. Completeness of Asynchronous Session Tree Subtyping in Coq. In *ITP (LIPICS)*, Yves Bertot, Temur Kutsia, and Michael Norrish (Eds.). Dagstuhl Publishing, Dagstuhl, 6:1–6:20. https://doi.org/10.4230/LIPIcs.ITP.2024.6

[18] Simon Gay and Malcolm Hole. 2005. Subtyping for Session Types in the pi-calculus. *Acta Informatica* 42, 2/3 (2005), 191–225. https://doi.org/10.1007/s00236-005-0177-z

[19] Silvia Ghilezan, Svetlana Jaksic, Jovanka Pantovic, Alceste Scalas, and Nobuko Yoshida. 2019. Precise Subtyping for Synchronous Multiparty Sessions. *J. Log. Algebraic Methods Program.* 104 (2019), 127–173. https://doi.org/10.1016/J.JLAMP.2018.12.002

[20] Silvia Ghilezan, Jovanka Pantovic, Ivan Prokic, Alceste Scalas, and Nobuko Yoshida. 2021. Precise Subtyping for Asynchronous Multiparty Sessions. *Proc. ACM Program. Lang.* 5, POPL (2021), 1–28. https://doi.org/10.1145/3434297

[21] Silvia Ghilezan, Jovanka Pantović, Ivan Prokić, Alceste Scalas, and Nobuko Yoshida. 2023. Precise Subtyping for Asynchronous Multiparty Sessions. *ACM Trans. Comput. Logic* 24, 2 (2023), 14:1–14:73. https://doi.org/10.1145/3568422

[22] Jonas Kastberg Hinrichsen. 2021. *Separations and Sessions*. PhD thesis. ITU. Available at https://jihgfee.github.io/.

[23] Kohei Honda. 1993. Types for Dyadic Interaction. In *CONCUR (LNCS, Vol. 715)*, Eike Best (Ed.). Springer, Heidelberg, 509–523.

[24] Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo. 1998. Language Primitives and Type Discipline for Structured Communication-based Programming. In *ESOP (LNCS, Vol. 1381)*, Chris Hankin (Ed.). Springer, Heidelberg, 122–138. https://doi.org/10.1007/BFb0053567

[25] Kohei Honda, Vasco Thudichum Vasconcelos, and Nobuko Yoshida. 2009. Type-Directed Compilation for Multicore Programming. *ENTCS* 241 (2009), 101–111. https://doi.org/10.1007/978-3-642-04167-9_12

[26] Kohei Honda, Nobuko Yoshida, and Marco Carbone. 2008. Multiparty Asynchronous Session Types. In *POPL*, George C. Necula and Philip Wadler (Eds.). ACM Press, New York, 273–284. https://doi.org/10.1145/1328897.1328472

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

[28] Hai Huang, Padmanabhan Pillai, and Kang G. Shin. 2002. Improving Wait-Free Algorithms for Interprocess Communication in Embedded Real-Time Systems. In *USENIX*, Carla Schlatter Ellis (Ed.). USENIX Association, Berkeley, 303–316. https://www.usenix.org/legacy/events/usenix02/huang.html

[29] Julien Lange and Nobuko Yoshida. 2017. On the Undecidability of Asynchronous Session Subtyping. In *FOSSACS (LNCS, Vol. 10203)*, Javier Esparza and Andrzej S. Murawski (Eds.). Springer, Heidelberg, 441–457. https://doi.org/10.1007/978-3-662-54458-7_26

[30] Jay Ligatti, Jeremy Blackburn, and Michael Nachtigal. 2017. On Subtyping-Relation Completeness, with an Application to Iso-Recursive Types. *ACM Trans. Program. Lang. Syst.* 39, 1 (2017), 4:1–4:36. https://doi.org/10.1145/2994596

[31] Dimitris Mostrous and Nobuko Yoshida. 2015. Session Typing and Asynchronous Subtyping for the Higher-order $\pi$-calculus. *Inf. Comput.* 241 (2015), 227–263. https://doi.org/10.1016/J.IC.2015.02.002

[32] Dimitris Mostrous, Nobuko Yoshida, and Kohei Honda. 2009. Global Principal Typing in Partially Commutative Asynchronous Sessions. In *ESOP (LNCS, 5502)*, Giuseppe Castagna (Ed.). Springer, Heidelberg, 316–332. https://doi.org/10.1007/978-3-642-00590-9_23

[33] Nicholas Ng, Jose G.F. Coutinho, and Nobuko Yoshida. 2015. Protocols by Default: Safe MPI Code Generation based on Session Types. In *CC (LNCS, Vol. 9031)*, Björn Franke (Ed.). Springer, Heidelberg, 212–232. https://doi.org/10.1007/978-3-662-46663-6_11

[34] Nicholas Ng and Nobuko Yoshida. 2015. Pabble: Parameterised Scribble. *Serv. Oriented Comput. Appl.* 9(3-4) (2015), 269–284. https://doi.org/10.1007/s11761-014-0172-8

[35] Nicholas Ng, Nobuko Yoshida, and Kohei Honda. 2012. Multiparty Session C: Safe Parallel Programming with Message Optimisation. In *TOOLS (LNCS, Vol. 7304)*, Carlo A. Furia and Sebastian Nanz (Eds.). Springer, Heidelberg, 202–218. https://doi.org/10.1007/978-3-642-30561-0_15

[36] Kaku Takeuchi, Kohei Honda, and Makoto Kubo. 1994. An Interaction-based Language and its Typing System. In *PARLE (LNCS, Vol. 817)*, Chris Hankin (Ed.). Springer, Heidelberg, 122–138. https://doi.org/10.1007/BFb0053567

[37] Thien Udomsrirungruang and Nobuko Yoshida. 2024. Three Subtyping Algorithms for Binary Session Types and their Complexity Analyses. In *PLACES (EPTCS, Vol. 401)*, Diana Costa and Raymond Hu (Eds.). Open Publishing Association, Waterloo, 49–60. https://doi.org/10.4204/EPTCS.401.5

[38] Nobuko Yoshida, Vasco Thudichum Vasconcelos, Hervé Paulino, and Kohei Honda. 2008. Session-Based Compilation Framework for Multicore Programming. In *FMCO (LNCS, Vol. 5751)*, Frank S. de Boer, Marcello M. Bonsangue, and Eric Madelaine (Eds.). Springer, Heidelberg, 226–246. https://doi.org/10.1007/978-3-642-04167-9_12