The Expressiveness of Session Types

Jorge A. Pérez
University of Groningen, The Netherlands

It is fair to say that *expressiveness* has been key to the success and impact of session types: enhancing the expressivity of typed processes and/or the properties enforced by typing is often a strong motivation for developing new typed frameworks. Rigorously contrasting different variants of session types from an expressiveness perspective is a pressing and non-trivial challenge. This short note gives a unified overview of some of our works in this direction; they cover various angles, including: higher-order communication, binary and multiparty protocols, and "propositions-as-sessions". The talk at ST30 will reflect on these (recent) results but also on avenues for future exploration.

In a 2016 position paper [18], I argued about the challenge of *typed expressiveness* in concurrency. The challenge consists in lifting the techniques that have proved so successful for assessing the expressiveness of untyped π-calculi to the setting of (session) typed π-calculi. The motivations are similar as in the untyped case: we would like to understand the relative strengths of different type systems, and to compare them rigorously to understand their differences and relative merits. This is actually a dynamic problem, as the number of behavioral type systems appearing in the literature is ever increasing. By disciplining the use of channels/names in processes, (session) types strongly influence their expressiveness. Casting expressiveness studies in the typed setting puts the focus on the *correctness properties* ensured by typing: different type systems enforce different properties, and so they determine a new, crucial dimension for comparisons. This is relevant in order to have satisfactory 'apples-to-apples' comparisons.

We have pursued the challenge of typed expressiveness particularly within the project "Unifying Correctness for Communicating Software" (2019-2024), whose goal is to connect different type systems via new expressiveness results. The talk at ST30 will give a unified overview of some selected results:

Higher-Order Concurrency Session π-calculi with *higher-order communication*, in which exchanged values include processes, are interesting because they can specify code mobility. In [13, 15], we study a family of languages with higher-order communication. We identify a core higher-order session calculus, which supports only passing of abstractions (functions from names to processes) but lacks recursion. Still, it can correctly codify name passing, recursion, and higher-order abstractions (functions from processes to processes). We obtain a taxonomy of session-typed languages, defined by a series of *typed* encodability results: each result consists of a translation on processes (as usual) but also of a translation on types, which ensures that source protocols are properly translated. In all cases, types are essential in defining the translations and/or in proving their correctness. Defining this taxonomy requires (i) enhancing encodability criteria with session types and (ii) developing (typed) behavioral equivalences [14].

Minimal Session Types The study in [13, 15] considers a standard formulation of session types. In [3, 2], we explore a simpler formulation, which dispenses with sequencing at the level of types. This way, e.g., in a type such as '$!\langle S\rangle;T$' the continuation type T can only be 'end'. This formulation, dubbed *minimal* session types, resembles simple and linear types for the π-calculus but also the way in which (one-shot) channels are typically used in languages such as Go. Using the core higher-order session calculus from [13, 15], we prove that sequencing constructs in processes and session types is convenient but

Submitted to:
ST 30

redundant: every process typable with standard session types can be compiled down to a process typable with minimal session types. This shows that only sequentiality in processes is truly indispensable, as it can correctly codify sequentiality in types. This can be considered as a result of *absolute expressiveness*, which explains session types in terms of themselves, without appealing to extraneous type disciplines. The idea behind minimal session types is rather simple and yet robust, as the formulation and associated correctness results can be adapted also to sessions the first-order setting, as shown in [1].

Binary and Multiparty Session Types Arguably, the two most active research strands within session types concern *multiparty* session types and "*propositions-as-sessions*"—the propositions-as-types approach to session types. The work in [4] jointly addresses them by developing an analysis of multiparty protocols using the *binary* session type system derived from the Curry-Howard interpretation of linear logic as session types. The idea is to analyze a multiparty implementation (a collection of processes, one per protocol participant) together with a so-called *medium process*, which is synthesized from a given global type. Although our analysis is not an encodability result as usual, it does bear witness to the expressivity of the session types based on linear logic, and enables to transfer the deadlock-freedom for processes involving delegation and interleaving to the multiparty setting. The analysis has been generalized in [10], which develops a technique based on *router processes*, which "wrap" local implementations while enabling the composition of protocol participants in arbitrary topologies. Routers can be used in static verification (as shown in [10]) but also in run-time verification, as recently shown in [11].

Type Systems for Deadlock-Freedom The work [7, 8] compares two type systems that enforce the deadlock-freedom property: (i) a type system based on priorities, as pioneered by Kobayashi [12]; and (ii) a type system based on propositions-as-sessions. To enable a coherent, 'apples-to-apples' comparison, two classes of processes are defined, denoted $\mathscr{K}$ and $\mathscr{L}$, which contain the deadlock-free induced by the type systems of [12] and [19], respectively. One key result in [7, 8] is that $\mathscr{L} \subsetneq \mathscr{K}$, i.e., the priority-based approach to deadlock-freedom subsumes the proof-theoretical approach. A key insight is that while the class $\mathscr{L}$ contains exclusively processes with tree-like topologies, the class $\mathscr{K}$ contains also processes with safe forms of circular topologies. This work also gives translations of $\mathscr{K}$ into $\mathscr{L}$.

Typed Functions into Session-Typed Processes Encodings of the λ-calculus provide a significant stress test of the expressiveness of the π-calculus. Following this line, we have developed encodings of *resource* λ-calculi into session-typed π-calculi derived from propositions-as-sessions [16, 17]. A distinctive aspect here is *non-determinism*: in λ, non-determinism concerns fetching of resources from bags of available resources; in π, we have sessions that are non-deterministically available—they may be available but may also fail. Because our resource λ-calculi are equipped with intersection types, our encodability results delineate a new, surprising connection between intersection types and linear logic.

Beyond Linear Logic The discovery of 'propositions-as-sessions' has a substantial and direct impact on our research program on typed expressiveness, for two reasons. First, as the correspondence naturally entails several key correctness properties (fidelity, deadlock-freedom, confluence, strong normalization), it defines a *canonical class* of session processes, upon which all other classes should be compared. Second, logic suggests a principled avenue for extensions: new typed frameworks can be obtained by suitably extending the underlying (linear) logic. Representative examples of this kind of extensions include behavioral polymorphism [5] and domain-aware computation [6]; just right outside linear logic, recent work has established a new concurrent interpretation of the logic of *bunched implications* [9].

Acknowledgments I am grateful to all the co-authors of the works mentioned above: A. Arslanagic, L. Caires, O. Dardha, E. D'Osualdo, D. Frumin, B. van den Heuvel, D. Kouzapas, D. Nantes-Sobrinho, A-A. Palamariuc, J. Paulus, F. Pfenning, B. Toninho, E. Voogd, N. Yoshida.

References

[1] Alen Arslanagic, Anda-Amelia Palamariuc & Jorge A. Pérez (2021): *Minimal Session Types for the π-calculus*. In Niccolò Veltri, Nick Benton & Silvia Ghilezan, editors: *PPDP 2021: 23rd International Symposium on Principles and Practice of Declarative Programming, Tallinn, Estonia, September 6-8, 2021*, ACM, pp. 12:1–12:15, doi:10.1145/3479394.3479407. Available at `https://doi.org/10.1145/3479394.3479407`.

[2] Alen Arslanagic, Jorge A. Pérez & Dan Frumin (2023): *A Minimal Formulation of Session Types*. *CoRR* abs/2301.05301, doi:10.48550/arXiv.2301.05301. arXiv:2301.05301.

[3] Alen Arslanagic, Jorge A. Pérez & Erik Voogd (2019): *Minimal Session Types (Pearl)*. In Alastair F. Donaldson, editor: *33rd European Conference on Object-Oriented Programming, ECOOP 2019, July 15-19, 2019, London, United Kingdom*, *LIPIcs* 134, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 23:1–23:28, doi:10.4230/LIPIcs.ECOOP.2019.23. Available at `https://doi.org/10.4230/LIPIcs.ECOOP.2019.23`.

[4] Luís Caires & Jorge A. Pérez (2016): *Multiparty Session Types Within a Canonical Binary Theory, and Beyond*. In Elvira Albert & Ivan Lanese, editors: *Formal Techniques for Distributed Objects, Components, and Systems - 36th IFIP WG 6.1 International Conference, FORTE 2016, Held as Part of the 11th International Federated Conference on Distributed Computing Techniques, DisCoTec 2016, Heraklion, Crete, Greece, June 6-9, 2016, Proceedings*, *Lecture Notes in Computer Science* 9688, Springer, pp. 74–95, doi:10.1007/978-3-319-39570-8_6. Available at `https://doi.org/10.1007/978-3-319-39570-8_6`.

[5] Luís Caires, Jorge A. Pérez, Frank Pfenning & Bernardo Toninho (2013): *Behavioral Polymorphism and Parametricity in Session-Based Communication*. In Matthias Felleisen & Philippa Gardner, editors: *Programming Languages and Systems - 22nd European Symposium on Programming, ESOP 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, *Lecture Notes in Computer Science* 7792, Springer, pp. 330–349, doi:10.1007/978-3-642-37036-6_19. Available at `https://doi.org/10.1007/978-3-642-37036-6_19`.

[6] Luís Caires, Jorge A. Pérez, Frank Pfenning & Bernardo Toninho (2019): *Domain-Aware Session Types*. In Wan J. Fokkink & Rob van Glabbeek, editors: *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*, *LIPIcs* 140, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 39:1–39:17, doi:10.4230/LIPIcs.CONCUR.2019.39. Available at `https://doi.org/10.4230/LIPIcs.CONCUR.2019.39`.

[7] Ornela Dardha & Jorge A. Pérez (2015): *Comparing Deadlock-Free Session Typed Processes*. In Silvia Crafa & Daniel Gebler, editors: *Proceedings of the Combined 22th International Workshop on Expressiveness in Concurrency and 12th Workshop on Structural Operational Semantics, EXPRESS/SOS 2015, Madrid, Spain, 31st August 2015*, *EPTCS* 190, pp. 1–15, doi:10.4204/EPTCS.190.1. Available at `https://doi.org/10.4204/EPTCS.190.1`.

[8] Ornela Dardha & Jorge A. Pérez (2022): *Comparing type systems for deadlock freedom*. *J. Log. Algebraic Methods Program.* 124, p. 100717, doi:10.1016/j.jlamp.2021.100717. Available at `https://doi.org/10.1016/j.jlamp.2021.100717`.

[9] Dan Frumin, Emanuele D'Osualdo, Bas van den Heuvel & Jorge A. Pérez (2022): *A bunch of sessions: a propositions-as-sessions interpretation of bunched implications in channel-based concurrency*. *Proc. ACM Program. Lang.* 6(OOPSLA2), pp. 841–869, doi:10.1145/3563318. Available at `https://doi.org/10.1145/3563318`.

[10] Bas van den Heuvel & Jorge A. Pérez (2022): *A decentralized analysis of multiparty protocols*. *Sci. Comput. Program.* 222, p. 102840, doi:10.1016/j.scico.2022.102840. Available at `https://doi.org/10.1016/j.scico.2022.102840`.

[11] Bas van den Heuvel, Jorge A. Pérez & Rares A. Dobre (2023): *Monitoring Blackbox Implementations of Multiparty Session Protocols*. *CoRR* abs/2306.04204, doi:10.48550/arXiv.2306.04204. arXiv:2306.04204.

[12] Naoki Kobayashi (2006): *A New Type System for Deadlock-Free Processes*. In Christel Baier & Holger Hermanns, editors: *CONCUR 2006 - Concurrency Theory, 17th International Conference, CONCUR 2006, Bonn, Germany, August 27-30, 2006, Proceedings*, *Lecture Notes in Computer Science* 4137, Springer, pp. 233–247, doi:10.1007/11817949_16. Available at `https://doi.org/10.1007/11817949_16`.

[13] Dimitrios Kouzapas, Jorge A. Pérez & Nobuko Yoshida (2016): *On the Relative Expressiveness of Higher-Order Session Processes*. In Peter Thiemann, editor: *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, *Lecture Notes in Computer Science* 9632, Springer, pp. 446–475, doi:10.1007/978-3-662-49498-1_18. Available at `https://doi.org/10.1007/978-3-662-49498-1_18`.

[14] Dimitrios Kouzapas, Jorge A. Pérez & Nobuko Yoshida (2017): *Characteristic bisimulation for higher-order session processes*. *Acta Informatica* 54(3), pp. 271–341, doi:10.1007/s00236-016-0289-7. Available at `https://doi.org/10.1007/s00236-016-0289-7`.

[15] Dimitrios Kouzapas, Jorge A. Pérez & Nobuko Yoshida (2019): *On the relative expressiveness of higher-order session processes*. *Inf. Comput.* 268, doi:10.1016/j.ic.2019.06.002. Available at `https://doi.org/10.1016/j.ic.2019.06.002`.

[16] Joseph W. N. Paulus, Daniele Nantes-Sobrinho & Jorge A. Pérez (2021): *Non-Deterministic Functions as Non-Deterministic Processes*. In Naoki Kobayashi, editor: *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, July 17-24, 2021, Buenos Aires, Argentina (Virtual Conference)*, *LIPIcs* 195, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 21:1–21:22, doi:10.4230/LIPIcs.FSCD.2021.21. Available at `https://doi.org/10.4230/LIPIcs.FSCD.2021.21`.

[17] Joseph W. N. Paulus, Daniele Nantes-Sobrinho & Jorge A. Pérez (2021): *Types and Terms Translated: Unrestricted Resources in Encoding Functions as Processes*. In Henning Basold, Jesper Cockx & Silvia Ghilezan, editors: *27th International Conference on Types for Proofs and Programs, TYPES 2021, June 14-18, 2021, Leiden, The Netherlands (Virtual Conference)*, *LIPIcs* 239, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 11:1–11:24, doi:10.4230/LIPIcs.TYPES.2021.11. Available at `https://doi.org/10.4230/LIPIcs.TYPES.2021.11`.

[18] Jorge A. Pérez (2016): *The Challenge of Typed Expressiveness in Concurrency*. In Elvira Albert & Ivan Lanese, editors: *Formal Techniques for Distributed Objects, Components, and Systems - 36th IFIP WG 6.1 International Conference, FORTE 2016, Held as Part of the 11th International Federated Conference on Distributed Computing Techniques, DisCoTec 2016, Heraklion, Crete, Greece, June 6-9, 2016, Proceedings*, *Lecture Notes in Computer Science* 9688, Springer, pp. 239–247, doi:10.1007/978-3-319-39570-8_16. Available at `https://doi.org/10.1007/978-3-319-39570-8_16`.

[19] Philip Wadler (2012): *Propositions as sessions*. In: *Proceedings of the 17th ACM SIGPLAN international conference on Functional programming*, ICFP '12, Association for Computing Machinery, New York, NY, USA, pp. 273–286, doi:10.1145/2364527.2364568. Available at `https://doi.org/10.1145/2364527.2364568`.