Foundations of Session Types: 10 Years Later

Giuseppe Castagna
CNRS - IRIF
Université de Paris
France

Mariangiola Dezani-Ciancaglini
Dipartimento di Informatica
Università di Torino
Italy

Elena Giachino
Università di Bologna
Italy

Luca Padovani
Dipartimento di Informatica
Università di Torino
Italy

CCS CONCEPTS

• **Theory of computation → Process calculi**; **Type structures**; **Operational semantics**.

KEYWORDS

Concurrency, Communication-Centered Programming, Sessions Types, Testing Equivalencies, Semantic Subtyping

ACM Reference Format:
Giuseppe Castagna, Mariangiola Dezani-Ciancaglini, Elena Giachino, and Luca Padovani. 2019. Foundations of Session Types: 10 Years Later. In *Principles and Practice of Programming Languages 2019 (PPDP '19), October 7–9, 2019, Porto, Portugal.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3354166.3356340

We were thrilled to know that our PPDP'09 paper "Foundations of Session Types" [10] was selected for the PPDP Most Influential Paper 10-Year Award. Just moments after being notified of this, we couldn't help looking at the works that cited—and in some cases were inspired by—our own. The result is the following short note, in which we recollect the main ideas behind our own work and the related ones that followed. The tight gap between the award notification and the deadline for the production of the PPDP'19 proceedings prevent us from providing an exhaustive survey of the related literature and we apologize in advance for the conciseness of our report and any relevant omission. Fortunately, there exist recent surveys [2, 4, 14, 21] that may help the interested readers orient themselves into the vast literature of session types as a whole.

Sessions and session types have proved to be extremely successful concepts for the structuring and the analysis of communications in distributed systems. A *session* is a *private communication channel* through which participating processes, using the so-called session endpoints, can communicate without interference from other processes. This privacy property of sessions enables the modular reasoning on complex systems, whereby each session is treated—and *typed*—in isolation. Thus, it is relatively easy to conceive type systems that ensure *communication safety* (only messages of the expected type are exchanged), *protocol fidelity* (communication progresses according to a given sequence of input/output operations) and under some additional assumptions also *progress* (the system does not get stuck in a configuration with unperformed pending operations). These type systems associate each session endpoint with a *session type*, which essentially describes a communication protocol in terms of the sequence of operations a process is supposed to perform on that endpoint. Most often, this protocol description also contains branching points, whereby processes may choose (internally) or offer (externally) different continuations, depending on the value of a specific message that identifies a branch.

The milestone papers by Honda [17] and Honda et al. [18] introduce binary sessions and binary session types along with the key notion of *duality*. Duality captures the idea that peer endpoints of the same (binary) session should be used, according to the respective session types, in complementary ways: where one session type specifies the output of a message, its dual specifies the input of a message with the same type; where one session type specifies an internal choice, the dual one specifies an external choice. Ultimately, duality ensures that the interaction between two channels typed by dual session types will never get stuck.

Another notion that stems from type theory and that applies to session types is that of *subtyping*: it is safe to use a value with type T whereby a value with type S is expected, provided that T is a subtype of S. Gay and Hole [15] were the first to study a notion of subtyping for session types and their work has become a classic reference in the session type literature. They proceed by first defining a subtyping relation (in the particular case, using coinduction, since session types may be recursive), and then proving that this relation is sound (i.e., it does not compromise any of the safety properties ensured by sessions) if used in the type system for a particular process calculus.

Our goal in writing the PPDP'09 paper was to give a theory of session types that was as much language independent as possible and where all the relevant definitions, duality and subtyping in particular, were semantically justified. Our approach was founded on two key insights. The first one was the realization that duality and subtyping are closely connected. Laneve and Padovani [22] had already pinpointed remarkable connections between session subtyping and testing equivalences for processes [12], where the notion of duality between two session types can be generalized to the relation between a process and a test. The second key insight

PPDP '19, October 7–9, 2019, Porto, Portugal
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-7249-7/19/10…$15.00
https://doi.org/10.1145/3354166.3356340

was inspired by the work of Frisch et al. [13], who had shown that the notion of *subtyping* could emerge *semantically*, given a suitable interpretation of types. If a type is interpreted as a set of values, namely, as the set of all the possible results that the expressions of that type may produce, then subtyping can be defined as result containment: an expression of a given type T can be safely used in a context where an expression of type S is expected if and only if the results that may be produced by that expression are less than—i.e., included in—those that the context can safely handle. This explains why their approach is called *semantic subtyping*.

Transposing the notion of semantic subtyping from a traditional setting (where types describe values) to sessions (where types describe protocols) requires defining a set-theoretic interpretation of session types. Session types cannot be characterized by the set of their results, insofar as typical sessions do not produce any result but, rather, a set of possibly infinite interactions—more precisely, a set of possibly infinite sequences of interaction *offers*. In our PPDP'09 paper, the interpretation of a session type is taken to be the set of its duals and two session types are related by subtyping if and only if all dual types of the smaller type were dual of the larger type as well. This construction is reminiscent of the realizability semantics of constructive logic. In hindsight, this interpretation is rather natural recalling that subtyping is a preorder relation defined to capture "safe replacement": every expression of a given type can be safely (in the sense of Wright and Felleisen [29]) used wherever an expression of a supertype is expected. Compared to the work on semantic subtyping by Frisch et al. [13], in PPDP'09 we skip one step, so to speak: instead of characterizing a type by a set of results and, indirectly, a context by the set of results it can safely handle, we directly characterized a type by the set of contexts that can safely handle the interactions (i.e., the session type "results") offered by endpoints of that type. In light of that, it is then quite natural to interpret a session type as the set of its dual session types, that is, as the set of contexts that can safely handle the interactions offered by that session type: this interpretation characterizes the interactions that an endpoint of that session type may offer (trivially, the more the dual types, the more the interactions the endpoint may produce). We thus recover the "safe replacement" interpretation of subtyping: an endpoint implementing a session type T can be safely used in a context where an endpoint of a session type S is expected if and only if the interactions it may offer are less than—i.e., included in—those that the context can safely handle. Our transposition of semantic subtyping to session types has subsequently inspired the subtyping relations given by Bartoletti et al. [5] and Barbanera and de'Liguoro [3]. Bejleri et al. [6] also model internal and external choices according to the same intuition.

One of the most intriguing aspects of the PPDP'09 semantic approach to the definition of session subtyping is the relationship between the internal and external choice operators and the intersection and union connectives. It turns out that, even though intersection and internal choice play different roles, the former being a pattern descriptor and the latter a behavioral operator, their semantics coincide. This does not hold for union and external choice however, since outputs in external choices behave like in internal ones. Union and external choice instead perfectly match in the simplified calculus of Padovani [25]. In the framework of the logical interpretation of session types [9], Acay and Pfenning [1] represent internal and external choices by intersection and union, respectively.

A technical challenge of our approach is that the semantic interpretation of session types is given by a labeled transition system which uses the notion of duality. In turn, duality is defined by means of the session type interpretation. We address this circularity by means of a suitable stratification of session types. A similar approach is adopted by Bono and Padovani [8], Barbanera and de'Liguoro [3] and Hüttel [20], while the labeled transition system of Bartoletti et al. [5] is first-order and populated by both processes and types.

As an application of our theory, the PPDP'09 paper presents a session type system for π-calculus processes where choices are based on the type of exchanged messages, instead of relying on labels. Notoriously, it is not trivial to guarantee the absence of deadlock when processes are allowed to interact through more than one session. Our solution makes use of a stack to constrain the order in which sessions can be used: only the session on top of the stack can communicate. Similar approaches to prevent deadlocks are described by Padovani [24] and Spaccasassi and Koutavas [27, 28]. Subsequently, other approaches to prevent deadlocks and not requiring an explicit ordering of sessions have emerged following the logical interpretation of session types [9].

Related to the notion of subtyping is that of *preciseness* [23], which requires to find a context expecting a term of type T and a term of type S such that filling the context with the term "goes wrong" whenever S is not a subtype of T. Preciseness for session types is studied by Chen et al. [11] and Ghilezan et al. [16] referring to our PPDP'09 paper, since semantic subtyping is precise by construction.

We conclude this note observing that two problems left open by Castagna et al. [10] have now been solved. The first one is to allow a channel to transmit a message containing itself, and in particular to transmit just itself. Bernardi and Hennessy [7] give a set theoretic model of session types without the need of stratification, so that it becomes possible to type channels carrying themselves. The second problem is the extension of the approach to multiparty session types [19], which describe interactions between a fixed but arbitrary number of processes. This extension has been developed by Padovani [26], where session types can be composed also using the parallel operator.

REFERENCES

[1] Cosku Acay and Frank Pfenning. 2016. Intersections and Unions of Session Types. In *Proceedings Eighth Workshop on Intersection Types and Related Systems, ITRS 2016, Porto, Portugal, 26th June 2016. (EPTCS)*, Naoki Kobayashi (Ed.), Vol. 242. Open Publishing Association, Waterloo, Australia, 4–19. https://doi.org/10.4204/EPTCS.242.3

[2] Davide Ancona, Viviana Bono, Mario Bravetti, Joana Campos, Giuseppe Castagna, Pierre-Malo Deniélou, Simon J. Gay, Nils Gesbert, Elena Giachino, Raymond Hu, Einar Broch Johnsen, Francisco Martins, Viviana Mascardi, Fabrizio Montesi, Rumyana Neykova, Nicholas Ng, Luca Padovani, Vasco T. Vasconcelos, and Nobuko Yoshida. 2016. Behavioral Types in Programming Languages. *Foundations and Trends in Programming Languages* 3, 2-3 (2016), 95–230. https://doi.org/10.1561/2500000031

[3] Franco Barbanera and Ugo de'Liguoro. 2015. Sub-behaviour Relations for Session-Based Client/Server Systems. *Mathematical Structures in Computer Science* 25, 6 (2015), 1339–1381. https://doi.org/10.1017/S096012951400005X

[4] Massimo Bartoletti, Ilaria Castellani, Pierre-Malo Deniélou, Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, Jovanka Pantovic, Jorge A. Pérez, Peter Thiemann, Bernardo Toninho, and Hugo Torres Vieira. 2015. Combining Behavioural Types with Security Analysis. *J. Log. Algebr. Meth. Program.* 84, 6 (2015), 763–780.

https://doi.org/10.1016/j.jlamp.2015.09.003

[5] Massimo Bartoletti, Alceste Scalas, and Roberto Zunino. 2014. A Semantic Deconstruction of Session Types. In *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings (Lecture Notes in Computer Science)*, Paolo Baldan and Daniele Gorla (Eds.), Vol. 8704. Springer, New York, NY, 402–418. https://doi.org/10.1007/978-3-662-44584-6_28

[6] Andi Bejleri, Elton Domnori, Malte Viering, Patrick Eugster, and Mira Mezini. 2019. Comprehensive Multiparty Session Types. *Programming Journal* 3, 3 (2019), 6. https://doi.org/10.22152/programming-journal.org/2019/3/6

[7] Giovanni Bernardi and Matthew Hennessy. 2016. Using Higher-Order Contracts to Model Session Types. *Logical Methods in Computer Science* 12, 2 (2016), 1–42. https://doi.org/10.2168/LMCS-12(2:10)2016

[8] Viviana Bono and Luca Padovani. 2012. Typing Copyless Message Passing. *Logical Methods in Computer Science* 8, 1 (2012), 1–50. https://doi.org/10.2168/LMCS-8(1:17)2012

[9] Luís Caires and Frank Pfenning. 2010. Session Types as Intuitionistic Linear Propositions. In *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings (Lecture Notes in Computer Science)*, Paul Gastin and François Laroussinie (Eds.), Vol. 6269. Springer, New York, NY, 222–236. https://doi.org/10.1007/978-3-642-15375-4_16

[10] Giuseppe Castagna, Mariangiola Dezani-Ciancaglini, Elena Giachino, and Luca Padovani. 2009. Foundations of Session Types. In *Proceedings of the 11th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, September 7-9, 2009, Coimbra, Portugal*, António Porto and Francisco Javier López-Fraguas (Eds.). ACM, New York, NY, 219–230. https://doi.org/10.1145/1599410.1599437

[11] Tzu-Chun Chen, Mariangiola Dezani-Ciancaglini, Alceste Scalas, and Nobuko Yoshida. 2017. On the Preciseness of Subtyping in Session Types. *Logical Methods in Computer Science* 13, 2 (2017), 1–61. https://doi.org/10.23638/LMCS-13(2:12)2017

[12] Rocco De Nicola and Matthew Hennessy. 1984. Testing Equivalences for Processes. *Theor. Comput. Sci.* 34 (1984), 83–133. https://doi.org/10.1016/0304-3975(84)90113-0

[13] Alain Frisch, Giuseppe Castagna, and Véronique Benzaken. 2008. Semantic Subtyping: Dealing Set-Theoretically with Function, Union, Intersection, and Negation Types. *J. ACM* 55, 4 (2008), 19:1–19:64. https://doi.org/10.1145/1391289.1391293

[14] Simon Gay and António Ravara (Eds.). 2017. *Behavioural Types: from Theory to Tools*. River Publishers, Gistrup, Denmark. https://doi.org/10.13052/rp-9788793519817

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

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

[17] Kohei Honda. 1993. Types for Dyadic Interaction. In *CONCUR '93, 4th International Conference on Concurrency Theory, Hildesheim, Germany, August 23-26, 1993, Proceedings (Lecture Notes in Computer Science)*, Eike Best (Ed.), Vol. 715. Springer, New York, NY, 509–523. https://doi.org/10.1007/3-540-57208-2_35

[18] Kohei Honda, Vasco Thudichum Vasconcelos, and Makoto Kubo. 1998. Language Primitives and Type Discipline for Structured Communication-Based Programming. In *Programming Languages and Systems - ESOP'98, 7th European Symposium on Programming, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'98, Lisbon, Portugal, March 28 - April 4, 1998, Proceedings (Lecture Notes in Computer Science)*, Chris Hankin (Ed.), Vol. 1381. Springer, New York, NY, 122–138. https://doi.org/10.1007/BFb0053567

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

[20] Hans Hüttel. 2016. Binary Session Types for Psi-Calculi. In *Programming Languages and Systems - 14th Asian Symposium, APLAS 2016, Hanoi, Vietnam, November 21-23, 2016, Proceedings (Lecture Notes in Computer Science)*, Atsushi Igarashi (Ed.), Vol. 10017. Springer, New York, NY, 96–115. https://doi.org/10.1007/978-3-319-47958-3_6

[21] Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luís Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, António Ravara, Emilio Tuosto, Hugo Torres Vieira, and Gianluigi Zavattaro. 2016. Foundations of Session Types and Behavioural Contracts. *ACM Comput. Surv.* 49, 1 (2016), 3:1–3:36. https://doi.org/10.1145/2873052

[22] Cosimo Laneve and Luca Padovani. 2008. The Pairing of Contracts and Session Types. In *Concurrency, Graphs and Models, Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday (Lecture Notes in Computer Science)*, Pierpaolo Degano, Rocco De Nicola, and José Meseguer (Eds.), Vol. 5065. Springer, New York, NY, 681–700. https://doi.org/10.1007/978-3-540-68679-8_42

[23] 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

[24] Luca Padovani. 2009. Session Types at the Mirror. In *Proceedings 2nd Interaction and Concurrency Experience: Structured Interactions, ICE 2009, Bologna, Italy, 31st August 2009. (EPTCS)*, Filippo Bonchi, Davide Grohmann, Paola Spoletini, and Emilio Tuosto (Eds.), Vol. 12. Open Publishing Association, Waterloo, Australia, 71–86. https://doi.org/10.4204/EPTCS.12.5

[25] Luca Padovani. 2010. Session Types = Intersection Types + Union Types. In *Proceedings Fifth Workshop on Intersection Types and Related Systems, ITRS 2010, Edinburgh, U.K., 9th July 2010. (EPTCS)*, Elaine Pimentel, Betti Venneri, and Joe B. Wells (Eds.), Vol. 45. Open Publishing Association, Waterloo, Australia, 71–89. https://doi.org/10.4204/EPTCS.45.6

[26] Luca Padovani. 2012. On Projecting Processes into Session Types. *Mathematical Structures in Computer Science* 22, 2 (2012), 237–289. https://doi.org/10.1017/S0960129511000405

[27] Carlo Spaccasassi and Vasileios Koutavas. 2015. Complete Session Types Inference with Progress Guarantees for ML. *CoRR* abs/1510.03929 (2015), 1–30. arXiv:1510.03929 http://arxiv.org/abs/1510.03929

[28] Carlo Spaccasassi and Vasileios Koutavas. 2016. Type-Based Analysis for Session Inference (Extended Abstract). In *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)*, Elvira Albert and Ivan Lanese (Eds.), Vol. 9688. Springer, New York, NY, 248–266. https://doi.org/10.1007/978-3-319-39570-8_17

[29] Andrew K. Wright and Matthias Felleisen. 1994. A Syntactic Approach to Type Soundness. *Inf. Comput.* 115, 1 (1994), 38–94. https://doi.org/10.1006/inco.1994.1093