# FuSe: A Simple Library Implementation of Binary Sessions
**(tool demonstration)**

Hernán Melgratti (Buenos Aires, Argentina) and Luca Padovani (Torino, Italy)

Gay and Vasconcelos [2010] have defined a session type system that integrates smoothly with an ML-like functional language. From a practical viewpoint, however, the type system appears challenging to adopt in a mainstream programming language, for it requires peculiar features for (F.1) describing *structured protocols* as sequences of I/O operations and branching points, (F.2) checking that the peer endpoints of a session are used according to *dual* protocols, and (F.3) ensuring the *linear usage* of session endpoints. Successful attempts to encode these features in Haskell have been reported by Neubauer and Thiemann [2004], Sackman and Eisenbach [2008], Pucella and Tov [2008], and Imai et al. [2010]. These works accomplish (F.1) and (F.2) using advanced features of Haskell's type system and (F.3) by encapsulating endpoints in a suitable monad. As discussed by Imai et al. [2010], however, all of these solutions have a price in terms of expressiveness, usability, or portability.

FuSe [Padovani, 2015] is an OCaml implementation of binary sessions that diverges from previous approaches in two ways. Concerning (F.1), FuSe represents session types according to the continuation-passing encoding of binary sessions documented by Dardha et al. [2012] while maintaining the natural communication semantics given by Gay and Vasconcelos [2010]. The main advantage of the chosen representation of session types is that duality boils down to type equality and this gives a straightforward and portable solution to (F.2). Concerning (F.3), FuSe adopts a lightweight mechanism that detects potentially unsafe linearity violations at runtime. Similar mechanisms have been described by Tov and Pucella [2010] and Hu and Yoshida [2016]. With this setup and piggybacking on OCaml, FuSe provides a faithful implementation of the communication API given by Gay and Vasconcelos [2010]. In addition, it supports *equi-recursive* and *polymorphic* session types [Bono et al., 2013], session *subtyping* [Gay and Hole, 2005], and enables *complete session type inference* through OCaml's inference engine.

In this demonstration we glance at the internals of FuSe and then see it at work on a few well-known examples taken from the existing literature on session types. We devote part of the demonstration to illustrate the effectiveness of FuSe in detecting errors and providing comprehensible inferred types and error messages. In particular, we argue that the very nature of session types makes it possible to detect a fair number of linearity violations even if OCaml's type system is not sub-structural. To compensate for the fact that encoded session types are often difficult to comprehend, FuSe provides an accompanying utility called `rosetta` that pretty prints OCaml types using the familiar session type notation.

Two more features of FuSe are worth mentioning, but their demonstration is subjected to time availability. Recently, Thiemann and Vasconcelos [2016] have introduced *context-free session types* to enable precise descriptions of protocols that elude the expressiveness of conventional session types. Examples include serialization protocols for tree-like data structures and XML documents and interaction protocols with non-uniform objects [Nierstrasz, 1993, Ravara and Vasconcelos, 2000] such as buffers and reentrant locks. FuSe provides a practical solution to the problems of context-free session type checking (left open by Thiemann and Vasconcelos) and context-free session type inference as well. Finally, we have developed a theory of *contract monitoring* [Findler and Felleisen, 2002] adapting to sessions the concept of *chaperone contract* [Strickland et al., 2012] and proving a *blame theorem* result [Wadler and Findler, 2009] in presence of possibly dependent session contracts. Runtime contract monitoring has been implemented and will be made publicly available in a forthcoming relase of FuSe.

Submitted to:
BETTY Final Meeting - 6-7 October 2016

## References

Viviana Bono, Luca Padovani, and Andrea Tosatto. Polymorphic Types for Leak Detection in a Session-Oriented Functional Language. In *Proceedings of FORTE'13*, LNCS 7892, pages 83–98. Springer, 2013.

Ornela Dardha, Elena Giachino, and Davide Sangiorgi. Session types revisited. In *Proceedings of PPDP'12*, pages 139–150. ACM, 2012.

Robert Bruce Findler and Matthias Felleisen. Contracts for higher-order functions. In *Proceedings of ICFP'02*, pages 48–59. ACM, 2002.

Simon Gay and Malcolm Hole. Subtyping for Session Types in the $\pi$-calculus. *Acta Informatica*, 42 (2-3):191–225, 2005.

Simon J. Gay and Vasco Thudichum Vasconcelos. Linear type theory for asynchronous session types. *Journal of Functional Programming*, 20(1):19–50, 2010.

Raymond Hu and Nobuko Yoshida. Hybrid Session Verification through Endpoint API Generation. In *Proceedings of FASE'16*, LNCS 9633, pages 401–418. Springer, 2016.

Keigo Imai, Shoji Yuen, and Kiyoshi Agusa. Session Type Inference in Haskell. In *Proceedings of PLACES'10*, EPTCS 69, pages 74–91, 2010.

Matthias Neubauer and Peter Thiemann. An implementation of session types. In *Proceedings of PADL'04*, LNCS 3057, pages 56–70. Springer, 2004.

Oscar Nierstrasz. Regular types for active objects. In *Proceedings of OOPSLA'93*, pages 1–15. ACM, 1993.

Luca Padovani. A Simple Library Implementation of Binary Sessions. Technical report, 2015. Document available at `https://hal.archives-ouvertes.fr/hal-01216310`, source code and tutorial available at `http://www.di.unito.it/~padovani/Software/FuSe/FuSe.html`.

Riccardo Pucella and Jesse A. Tov. Haskell session types with (almost) no class. In *Proceedings of HASKELL'08*, pages 25–36. ACM, 2008.

António Ravara and Vasco Thudichum Vasconcelos. Typing non-uniform concurrent objects. In *Proceedings of CONCUR'00*, LNCS 1877, pages 474–488. Springer, 2000.

Matthew Sackman and Susan Eisenbach. Session Types in Haskell: Updating Message Passing for the 21st Century. Technical report, Imperial College London, 2008. Available at `http://pubs.doc.ic.ac.uk/session-types-in-haskell/`.

T. Stephen Strickland, Sam Tobin-Hochstadt, Robert Bruce Findler, and Matthew Flatt. Chaperones and impersonators: run-time support for reasonable interposition. In *Proceedings of OOPSLA'12*, pages 943–962. ACM, 2012.

Peter Thiemann and Vasco T. Vasconcelos. Context-Free Session Types. In *Proceedings of ICFP'16*, pages 462–475. ACM, 2016.

Jesse A. Tov and Riccardo Pucella. Stateful Contracts for Affine Types. In *Proceedings of ESOP'10*, LNCS 6012, pages 550–569. Springer, 2010.

Philip Wadler and Robert Bruce Findler. Well-typed programs can't be blamed. In *Proceedings of ESOP'09*, LNCS 5502, pages 1–16. Springer, 2009.