Interleaving Sessions with Predicates

Peter Thiemann
Albert-Ludwigs-Universität Freiburg
Freiburg, Germany
thiemann@acm.org

ABSTRACT

We propose an extension of binary session types with interleaved sessions that work similar to interrupts. The participants in a session register a set of trigger predicates that are associated with local protocols. When a trigger fires, all participants switch simultaneously to the interleaved protocol and return to the original protocol afterwards. The use of interleaved sessions enables the elegant specification of protocols that are otherwise cumbersome to specify.

1. INTRODUCTION

Session types enable fine-grained static control over communication protocols. They evolved from a structuring device for two-party communication in π-calculus [Honda, 1993] (binary session types) over calculi embedded in functional languages [Gay and Hole, 1999, Vasconcelos et al., 2006] to a powerful means for describing multi-party orchestration of communication [Honda et al., 2008]. There are embeddings in object-oriented languages [Gay et al., 2010, Dezani-Ciancaglini et al., 2009]. Their logical foundations have been investigated with interpretations in intuitionistic and classical linear logic [Caires and Pfenning, 2010, Wadler, 2012].

However, most uses of session types deal with a single protocol that does not change while the system is running. One exception are the systems for session types with exceptions by Carbone and coworkers [Carbone et al., 2009]. They propose a system (that we call CHY) where session types are extended with exceptional session handlers. A globally visible event causes all participants of a session to switch protocol to the handler corresponding to the event.

Another system that changes protocols on the go is the adaptive system for secure information flow by Castellani and coworkers [Castellani et al., 2016]. In this system, the communication partners switch protocols in response to an observed breach of a security property. The switch is anticipated in the type structure with union and intersection types.

Capecchi and coworkers [Capecchi et al., 2016] propose a mechanism for multi-party sessions that throws an exception to a subset of the participants in the session in a limited scope. Their framework relies on the broadcasting of an exception message.

Protocol types [Chen et al., 2016] may be viewed as a refinement of Capecchi's work where exceptions are tied to single exchanges and further distinguished into different failures that lead to different handler protocols.

Demangeon and coworkers [Demangeon et al., 2015] consider run-time verification for multi-party session types with scoped exceptions. They put particular emphasis on asynchrony such that messages in a scope with a thrown exception are skipped until all participants have become aware of the exception.

In contrast, we consider statically enforced protocols that are interleaved just like interrupt handlers are interleaved with the normal execution of a program: after handling the interrupt, normal execution is resumed. None of the previously discussed works allows for resumption. In our approach, each session (endpoint) is associated with a state that evolves in a loosely synchronized way. Handlers can be associated with predicates on the session state. If a predicate holds on the current session state (following a state update or a communication), then this fact is observed by all participants of the protocol and they switch to the interleaving protocol to handle the interrupt.

Interleaved sessions return to where they left off in the original protocol, while the systems discussed above (e.g., CHY) discard the rest of the protocol inside the scope of the exception handler and continue outside after handling the exception. The next section gives a number of applications for interleaved sessions.

We define the formal system, give its semantics by embedding it into a variant of Gay and Vasconcelos' system of functional session types restricted to first-order sessions and synchronous communication, and sketch a proof of type preservation. We believe this framework makes it easy to discuss realistic examples and it simplifies the metatheory because the functional part yields a simple way to define and manipulate session state separate from the communication operations.

Appears in proceedings SAC 2017

2. REFERENCES

[Caires and Pfenning, 2010] Caires, L. and Pfenning, F. (2010). Session types as intuitionistic linear propositions. In *CONCUR*, volume 6269 of *LNCS*, pages 222–236, Paris, France. Springer.

[Capecchi et al., 2016] Capecchi, S., Giachino, E., and Yoshida, N. (2016). Global escape in multiparty sessions. *Mathematical Structures in Computer Science*, 26(2):156–205.

[Carbone et al., 2009] Carbone, M., Yoshida, N., and Honda, K. (2009). Asynchronous session types: Exceptions and multiparty interactions. In Bernardo, M., Padovani, L., and Zavattaro, G., editors, *SFM*, volume 5569 of *LNCS*, pages 187–212. Springer.

[Castellani et al., 2016] Castellani, I., Dezani-Ciancaglini, M., and Pérez, J. A. (2016). Self-adaptation and secure information flow in multiparty communications. *Formal Asp. Comput.*, 28(4):669–696.

[Chen et al., 2016] Chen, T., Viering, M., Bejleri, A., Ziarek, L., and Eugster, P. (2016). A type theory for robust failure handling in distributed systems. In *FORTE*, volume 9688 of *LNCS*, pages 96–113. Springer.

[Demangeon et al., 2015] Demangeon, R., Honda, K., Hu, R., Neykova, R., and Yoshida, N. (2015). Practical interruptible conversations: Distributed dynamic verification with multiparty session types and Python. *Formal Methods in System Design*, 46(3):197–225.

[Dezani-Ciancaglini et al., 2009] Dezani-Ciancaglini, M., Drossopoulou, S., Mostrous, D., and Yoshida, N. (2009). Objects and session types. *Information and Computation*, 207(5):595–641.

[Gay and Hole, 1999] Gay, S. J. and Hole, M. J. (1999). Types and subtypes for client-server interactions. In *Proc. 1999 ESOP*, volume 1576 of *LNCS*, pages 74–90, Amsterdam, The Netherlands. Springer.

[Gay et al., 2010] Gay, S. J., Vasconcelos, V. T., Ravara, A., Gesbert, N., and Caldeira, A. Z. (2010). Modular session types for distributed object-oriented programming. In *Proc. 37th ACM Symp. POPL*, pages 299–312, Madrid, Spain. ACM Press.

[Honda, 1993] Honda, K. (1993). Types for dyadic interaction. In Best, E., editor, *Proceedings of 4th International Conference on Concurrency Theory*, number 715 in LNCS, pages 509–523.

[Honda et al., 2008] Honda, K., Yoshida, N., and Carbone, M. (2008). Multiparty asynchronous session types. In Wadler, P., editor, *Proc. 35th ACM Symp. POPL*, pages 273–284, San Francisco, CA, USA. ACM Press.

[Vasconcelos et al., 2006] Vasconcelos, V. T., Ravara, A., and Gay, S. J. (2006). Type checking a multithreaded functional language with session types. *Theoretical Computer Science*, 368(1-2):64–87.

[Wadler, 2012] Wadler, P. (2012). Propositions as sessions. In Findler, R. B., editor, *ICFP'12*, pages 273–286, Copenhagen, Denmark. ACM.