Reactive Sessions

Mauricio Cano
University of Groningen, The Netherlands
http://www.mcanog.info

Jorge A. Pérez
University of Groningen, The Netherlands
http://www.jperez.nl

Widely studied programming models for communication-based systems, such as the π-calculus [7], tend to be overly rigid, in that they do not naturally capture the reactive behavior of such software systems. Reactive behavior is critical for modern autonomous agents which can automatically engage in communication protocols in our behalf (e.g. financial transactions). In communication-based systems, reactive behavior encompasses several features and constructs, including time, exception handling and dynamic reconfiguration. Although all of these features have been studied as (isolated) extensions/variants of the session π-calculus (cf. [3, 2]), the resulting models are often convoluted, which limits its potential for reasoning about communicating systems.

Given the relevance of reactive behavior, in ongoing work we have been exploring a fresh look at programming models for communication-based concurrency. We have focused on the family of synchronous programming languages (SRP) [1], as a foundational model for communication-based systems. SRP is an event-based model of computation optimized for programming reactive systems. It is based on the hypothesis of perfect synchrony: reactive programs react instantaneously and produce their outputs synchronously with their input. An SRP program evolves deterministically as an (infinite) sequence of reactions, indexed by a logical clock.

Our work focuses on ReactiveML, a particular synchronous programming language supported on formal foundations [6, 5]. In ReactiveML inputs are defined as signals, which can be emitted at any time in execution. Time units correspond to a series of internal computations, whose execution is stopped when there are no further signals to be emitted. A distinguishing feature of ReactiveML is that reactions to the absence of signals only take place at the end of the time unit.

We have encoded the basic structures of session-based communication (e.g. input/output, branch/selection) into ReactiveML. This encoding is based on two main notions: (1) each synchronization corresponds to a single time unit, and (2) a disciplined use of a continuation-passing style representation of ses-

sions (following [4]) enables to represent the linearity and polymorphism of communication channels. An associated operational correspondence result formally guarantees that the communication primitives are modeled correctly in ReactiveML.

The talk will present the current status of our work on session-based concurrency in ReactiveML. We will also discuss directions for future work, aimed at us- ing the encoding (and its ReactiveML implementation) to exploit important verification techniques (e.g., type-checking, monitors, runtime checking). The ultimate goal is to certify correctness in scenarios in which reactivity plays an important role in structured communications, such as the hot-swapping of web-services and modules.

References

[1] A. Benveniste, P. Caspi, S. A. Edwards, N. Halbwachs, P. L. Guernic, and R. de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1):64–83, 2003.

[2] L. Bocchi, W. Yang, and N. Yoshida. Timed multiparty session types. In *Proc. of CONCUR'14*, volume 8704, pages 419–434. Springer, 2014.

[3] M. Carbone. Session-based choreography with exceptions. *Electr. Notes Theor. Comput. Sci.*, 241:35–55, 2009.

[4] O. Dardha, E. Giachino, and D. Sangiorgi. Session types revisited. In *Proc. of PPDP'12*, pages 139–150, 2012.

[5] L. Mandel and C. Pasteur. Reactivity of Cooperative Systems - Application to ReactiveML. In M. Müller-Olm and H. Seidl, editors, *Static Analysis - 21st International Symposium, SAS 2014, Munich, Germany, September 11-13, 2014. Proceedings*, volume 8723 of *Lecture Notes in Computer Science*, pages 219–236. Springer, 2014.

[6] L. Mandel and M. Pouzet. ReactiveML: a reactive extension to ML. In *Proc. of PPDP'05*, pages 82–93. ACM, 2005.

[7] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I. *Inf. Comput.*, 100(1):1–40, 1992.