Session Types: Achievements and Challenges

Simon J. Gay

School of Computing Science, University of Glasgow, UK
Simon.Gay@glasgow.ac.uk

Abstract

Session types are type-theoretic specifications of communication protocols, introduced by Kohei Honda and collaborators in the mid-1990s. They define the type and sequence of messages exchanged via a communication medium, and allow type-checking techniques to be used to verify protocol implementations. Whereas data types codify the static structure of information in a computer program, session types codify the dynamic structure of communication in a software system. The classic slogan *"algorithms + data structures = programs"* can be generalised to *"programs + communication structures = systems"*, and the full range of type-checking technology can be generalised too.

In the simplest form, a session type specifies a straightforward sequence of messages. The type $!\mathsf{int}.?\mathsf{bool}.\mathsf{end}$ describes how to run a protocol on an endpoint of a communication channel: first send (!) an integer, then receive (?) a boolean, then terminate. The other endpoint has the dual type $?\mathsf{int}.!\mathsf{bool}.\mathsf{end}$. More complex protocols include choice and repetition. For example, the recursive type S defined by $S = \&\langle \mathsf{start} : ?\mathsf{int}.!\mathsf{bool}.S, \mathsf{stop} : \mathsf{end}\rangle$ describes a protocol that offers a choice between start and stop, each with its own continuation protocol. The basic idea for protocol verification is to match the structure of a session type with the use of communication operations in a program.

The twenty years since the introduction of session types have seen a dramatic growth in research activity. There is now a substantial community, and most programming-language-related conferences regularly include papers on session types. Several themes of research can be identified:

- Generalisation of session types from two-party to multi-party sessions.
- Transfer of session types from pi-calculus to a range of programming language paradigms.
- Logical foundations of session types via a Curry-Howard correspondence with linear logic.
- Connections between session types and automata theory.
- Development of programming language implementations and session-type-based tools.
- Broadening the original focus on static type-checking to include dynamic monitoring.
- Incorporation of time and error-handling.
- Connections with more general type-theoretic concepts such as dependent types, gradual types and typestate.

The lecture will introduce session types, survey the main themes and achievements of the field, and suggest directions for future work that are likely to be of interest to researchers from the wider area of type theory.