
University of Glasgow | School of Computing Science

# Session types in the wild

A. Laura Voinea

First Supervisor: Dr. Simon Gay
Second Supervisor: Dr. Wim Vanderbauwhede

3nd Year PhD Report

June 13, 2018

# Contents

# 1 Introduction

Nowadays, distributed systems are ubiquitous and communication is an important feature and reason for its success. Communication-centred programming has proven to be one of the most successful attempts to replace shared memory for building concurrent, distributed systems. Communication is easier to reason about and scales well as opposed to shared memory, making it a more suitable approach for systems where scalability is a must, as in the case of multi-core programming, service-oriented applications or cloud computing[1].

Communication is usually standardised via protocols that specify the possible interactions between the communicating parties in a specific order. Mainstream programming languages fail to adequately support the development of communication-centred software. Thus, implementations of communication behaviours are often based on informal protocol specification, and so informal verification. As a result they are prone to errors such as communication mismatch, when the message sent by one party is not expected by the other party, or deadlock, when parties are waiting for a message from each other causing the system to block.[1].

Session types [9, 10, 12] are an established formalism for the enforcement of communication protocols through static analysis. Session types describe communication by specifying the type and direction of messages exchanged between parties[6]. Programmers can express a protocol specification as a session type, which can guarantee, within the scope where the session type applies, that communications will always match and the system will never deadlock.

The main goal of the ABCD project[1] is to improve the practice of software development for concurrent and distributed systems through the use of session types. This is to be accomplished through built-in language support for protocol codification in existing languages such as Java or Python, in new languages such as Links[2], inter-language interoperability via session types, and through adapting interactive development environments and modelling techniques to support session types. Logical and automata foundations of session types will be further developed to express a wider class of behaviour and, as need arises, to support the former. Empirical studies to assess methodologies and tools are to be carried out, with results being used to improve language and tool design and implementation.

## 1.1 Thesis Statement

Session types are a useful addition to the syntax and semantics of modern languages. Programming with session types and the constraints that this comes with i.e. linearity can be understood and used by real world programmers. Moreover session types can help programmers understand the problem they are tying to solve with more ease and structure their code better. Session typed languages provide useful additional safeguards and diagnostic information that lead to a system with the expected behaviour with less effort.

### 1.2 Outline

A short literature review of the most relevant work can be found in **??**.

Further work along with a plan for the time remaining is discussed in section **??**.

## 2 Future Work

### 2.1 Thesis Outline

1. Introduction
   1.1. Motivation and Objectives
   1.2. Contributions
   1.3. Publications
2. Background theory
   2.1. $\pi$-types
   2.2. Session types
3. Modular linearity with Capabilities
   3.1. Introduction
4. Modular linearity with other
   4.1. Introduction
5. Mungo and Paxos
   5.1. Introduction
6. Conclusion
   6.1. Summary of Thesis Achievements
   6.2. Future Work

| **Gantt chart** | 2018 | | | | | | | 2019 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 |
| Modular linearity with Capabilities | | | | | | | | | | | | | | | | |
| Semantic approach to modular linearity | | | | | | | | | | | | | | | | |
| Mungo | | | | | | | | | | | | | | | | |
| Thesis write-up | | | | | | | | | | | | | | | | |

## References

[1] A basis for concurrency and distribution. http://groups.inf.ed.ac.uk/abcd/.

[2] Links: Linking theory to practice for the web. http://groups.inf.ed.ac.uk/links/.