Formal Verification of Session Types

by

Théa Johnson,

Thesis

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Bachelor's of Arts

University of Dublin, Trinity College

April 2016

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Théa Johnson

April 8, 2016

Permission to Lend and/or Copy

Acknowledgments

...ACKNOWLEDGMENTS...

Théa Johnson

University of Dublin, Trinity College
April 2016

Formal Verification of Session Types

Théa Johnson, B.A.
University of Dublin, Trinity College, 2016

Supervisor: ...

...ABSTRACT...

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The aims of this project are to investigate the formal verification of session types, specifically in relation to the paper Type-Based Analysis for Session Inference [1], and to implement a behaviour checker based on the designs described in this paper.

1.1 Motivation

The modern world is growing increasingly dependent on distributed systems, changing the historical approach to computing dramatically. In order for modern society to function it is important that these systems communicate correctly and that when proposing or introducing new systems we can show that they will communicate correctly under all circumstances.

Modern programming languages support data types. These allow us to use verification techniques to show that the program will run as expected on all forms of input. A similar system could be used for communication over distributed systems. Ideally a type system for communication would be embedded into languages in a similar fashion to the type systems of all modern languages.

1.2 Current Work

This area is currently been researched by multiple groups. However it is currently not used in real world systems to any great extent. To date systems have been developed for applying session type disciplines to functional languages, object oriented languages and operating systems.

1.3 Background

Traditional type systems embedded into programming languages focus on the computations and what they should produces. Session type systems aim to embed session types that describe both the sequence and they type of messages that are been transmitted on communication channels into modern programming languages. This allows for the verification of the communications on a channel since the session type describes the protocol of the channel.

1.3.1 Main Session Type Approaches

According to Hüttle et al. [2] the main approaches to session types are as follows.

Session types are usually associated with binary communication channels where the two ends using the channel see it as complementary types. It is then possible to use static type checking to ensure that the communications on the channel are in accordance with the protocol. *Multiparty session types* extend binary session types to allow for more than two programs to communicate. *Conversation session types* unify local and global multipary types and allow for unspecified numbers for communication. *Contracts* focus on general theory to confirm that communications follow the specified abstract description of input/output actions.

Chapter 2

Summary of Paper Type Based Analysis for Session Inference

2.1 Overview

This paper introduces a new idea for a design approach to Binary Session Types which uses effects. A high level language is developed where communication protocols can be programmed and checked statically. In this paper [1] the approach is applied to ML_s, a core of the language ML with session communication.

The approach suggested separates traditional and session typing using a two level system. The first level uses a type and effect system, which is an adaptation of the one developed by Amtoft, Neilsen and Neilsen 1999 [3]. Types allow for clear representation of program properties and enable modular reasoning. The effects are essentially behaviour types that record the actions of interest that will happen at run time. From this system a complete behaviour inference algorithm is obtained, this extracts a behaviour for a program providing it respects ML types. In this level the programs are typed against both an ML type and a behaviour. Session protocols are not considered here and so endpoints are given a type ses^{ρ} instead (see 2.3).

The second level checks the behaviour to see that it complies with the session types of both channels and endpoints. In performing this check the operational semantics are used (see 2.3).

This level is inspired by the work done by Castagna et al. [4]. In their system session based interaction is established on a public channel, once established the parties continue to communicate on a private channel. Messages are exchanged on the private channel according to a given protocol. Internal and external choices are also required by this system to implement control. Internal choices are when the decision is made autonomously by a process and external choices occur when a decision is based entirely on messages received.

This level ensures that sessions respect the order of communication and message types described by the session type of the channel. It also ensures partial lock freedom due to stacked interleaving of sessions 2.3.

One of the most appealing aspects of the session type discipline proposed here is that it allows for complete session type inference from behaviours. When this is combined with behaviour inference from level 1 we get a method from complete session type inference without programmer annotations.

The two levels of the system only interact through behaviours. This allows for the development of front ends for different languages and back ends for different session disciplines and to combine the two to cover an extensive selection of requirements.

2.2 The Fist Level

At this level the type and effect system of Amtoft, Neilsen and Neilsen 1999 [3] is extended to session communications in ML_s. The type and effect system consists of constructions of judgments of the form $C; \Gamma \vdash e : T \triangleright b$. In this statement C represents the constraint environment which is used to relate type level variables to terms and so enables session inference. Γ represents the type environment which is used to bind program variables to type schemas. To read this judgment we would say that expression e has type T and behaviour b under type environment Γ and constraint environment C.

In the system designed in the paper an ML_s expression can have either a standard ML type or a session type. Session types are of the form ses^{ρ} where ρ is a static approximation of the location of the endpoint. Functional types have an associated behaviour β and type variables α are used for ML polymorphism.

Polymorphism is extended with type schemas. These are of the form $\forall(\overrightarrow{\gamma} : C_0).T$ where γ is a list made up of some combination of type (α), behaviour (β), region (ρ)

Figure 2.1: Syntax of types, behaviours, constraints and session types

Figure 2.2: Type and Effect system for ML_s Expressions omitting rule for pairs

and session (ψ) variables and C_0 represents the constraint environment that imposes constraints on the quantified variables.

The rules for the type and effect system proposed are given in fig. 2.2. These are made of of the judgments described above and requirements for the rule. These rules say that if we have a judgments of the form given above the line and if the requirements beside the rule (if they exist) are met then the judgment below the line should be valid.

2.3 The Second Level

Appendix

...

Bibliography

[1] C. Spaccasassi and V. Koutavas, “Type based analysis for session inference.”

[2] H. Hüttel, I. Lanese, V. T. Vasconcelos, L. Caires, M. Carbone, P.-M. Denilou, D. Mostrous, L. Padovani, A. Ravara, E. Tuosto, H. T. Vieira, and G. Zavattaro, “Foundations of behavioural types.” `http://www.behavioural-types.eu/publications`, 2014.

[3] T. Amtoft, F. Nielson, and H. R. Nielson, *Type and Effect System - Behaviours for Concurrency.* Imperial College Press, 1999.

[4] G. Castagna, M. Dezani-Ciancaglini, E. Giachino, and L. Padovani, “Foundations of session types,” 2009.