JFP **30**, e18, 7 pages, 2020. © The Author(s), 2020. Published by Cambridge University Press
doi:10.1017/S0956796820000167

PhD Abstracts

GRAHAM HUTTON
University of Nottingham, UK
(*e-mail:* graham.hutton@nottingham.ac.uk)

Many students complete PhDs in functional programming each year. As a service to the community, twice per year the Journal of Functional Programming publishes the abstracts from PhD dissertations completed during the previous year.

The abstracts are made freely available on the JFP website, that is, not behind any paywall. They do not require any transfer of copyright, merely a license from the author. A dissertation is eligible for inclusion if parts of it have or could have appeared in JFP, that is, if it is in the general area of functional programming. The abstracts are not reviewed.

We are delighted to publish six abstracts in this round and hope that JFP readers will find many interesting dissertations in this collection that they may not otherwise have seen. If a student or advisor would like to submit a dissertation abstract for publication in this series, please contact the series editor for further details.

Graham Hutton
PhD Abstract Editor

Engineering Self-Adaptive Collective Processes for Cyber-Physical Ecosystems

ROBERTO CASADEI
Alma Mater Studiorum-Università di Bologna, Italy

Date: April 2020; Advisor: Mirko Viroli
URL: https://tinyurl.com/y8dkcg7k

The pervasiveness of computing and networking is creating significant opportunities for building valuable socio-technical systems. However, the scale, density, heterogeneity, interdependence, and QoS constraints of many target systems pose severe operational and engineering challenges. Beyond individual smart devices, cyber-physical collectives can provide services or solve complex problems by leveraging a "system effect" while coordinating and adapting to context or environment change. Understanding and building systems exhibiting collective intelligence and autonomic capabilities represent a prominent research goal, partly covered, for example, by the field of collective adaptive systems. Therefore, drawing inspiration from and building on the long-time research activity on coordination, multi-agent systems, autonomic/self-$\star$ systems, spatial computing, and especially on the recent aggregate computing paradigm, this thesis investigates concepts, methods, and tools for the engineering of possibly large-scale heterogeneous ensembles of situated components that should be able to operate, adapt, and self-organize in a decentralized fashion. The primary contribution of this thesis consists of four main parts. First, we define and implement an aggregate programming language (ScaFi), internal to the mainstream Scala programming language, for describing collective adaptive behavior, based on field calculi. Second, we conceive of a "dynamic collective computation" abstraction, also called aggregate process, formalized by an extension to the field calculus, and implemented in ScaFi. Third, we characterize and provide a proof-of-concept implementation of a middleware for aggregate computing that enables the development of aggregate systems according to multiple architectural styles. Fourth, we apply and evaluate aggregate computing techniques to edge computing scenarios, and characterize a design pattern, called Self-organising Coordination Regions (SCR), that supports adjustable, decentralized decision-making and activity in dynamic environments.

Strategies for Testing and Formalizing Properties of Modern Programming Languages

SAMUEL DA SILVA FEITOSA
Federal University of Pelotas, Brazil

Date: December 2019; Advisor: Andre Rauber Du Bois, Rodrigo Geraldo Ribeiro and Wouter Swierstra
URL: `https://tinyurl.com/y7ejwsn5`

Today's world is full of devices and machines controlled by software, which depend upon programming languages and compilers to be produced and executed. The importance of correct software development goes beyond personal computers and smartphone apps. An error in a critical system, such as on a nuclear power plant or on an airplane controller, can cause catastrophic damage in our society. Nowadays, essentially two software validation techniques are used to avoid such problems: software testing and software verification.

In this thesis, we combine both validation techniques in the programming languages research area, applying property-based testing first to improve specifications and debugging programs, before an attempt of formal verification. Using such testing approach, we can quickly eliminate false conjectures, using the generated counterexamples, which help to correct them. Then, having confidence that the specification is correct, one can give a step forward and formalize the specification and prove its properties in an interactive theorem prover, which uses a mathematical framework to guarantee that these properties hold for a given specification.

We apply different strategies to test and formalize two major programming languages: the functional Lambda calculus and the modern object-oriented calculus Featherweight Java. The first branch of this thesis defines a type-directed procedure to generate random programs for each calculus in order to apply property-based testing to check soundness properties on them, using the Haskell library QuickCheck. And in the second branch, we apply the two most used approaches, extrinsic and intrinsic, to formalize and prove type safety for both studied programming languages using the dependently typed programming language Agda, comparing the subtleties of each technique. Furthermore, we show that our formalizations can be extended to new language constructions, by specifying and proving the same properties for Java 8 constructions. We believe that this combination of property-based testing with formal verification can improve the quality of software in general and increase productivity during formal proof development.

Effectful Programming in Declarative Languages with an Emphasis on Non-Determinism: Applications and Formal Reasoning

SANDRA DYLUS
Christian-Albrechts-University Kiel, Germany

Date: February 2020; Advisor: Michael Hanus
URL: https://tinyurl.com/y9amsj9v

This thesis investigates effectful declarative programming with an emphasis on non-determinism as an effect. On the one hand, we are interested in developing applications using non-determinism as underlying implementation idea. We discuss two applications using the functional logic programming language Curry. The key idea of these implementations is to exploit the interplay of non-determinism and non-strictness that Curry employs. The first application investigates sorting algorithms parametrized over a comparison function. By applying a non-deterministic predicate to these sorting functions, we gain a permutation enumeration function. We compare the implementation in Curry with an implementation in Haskell that uses a monadic interface to model non-determinism. The other application that we discuss in this work is a library for probabilistic programming. Instead of modeling distributions as list of event and probability pairs, we model distributions using Curry's built-in non-determinism. In both cases, we observe that the combination of non-determinism and non-strictness has advantages over an implementation using lists to model non-determinism. On the other hand, we present an idea to apply formal reasoning on effectful declarative programming languages. In order to start with simple effects, we focus on modeling a functional subset first. That is, the effects of interest are totality and partiality. We then observe that the general scheme to model these two effects can be generalized to capture a wide range of effects. Obviously, the next step is to apply the idea to model non-determinism. More precisely, we implement a model for the non-determinism of Curry: non-strict non-determinism with call-time choice. Therefore, we finally discuss why the current representation models call-by-name rather than Curry's call-by-need semantics and give an outlook on ideas to tackle this problem.

Narrowing in on Property-Based Testing

JON FOWLER
University of Nottingham, UK

Date: December 2019; Advisor: Graham Hutton
URL: https://tinyurl.com/y9dyxpvh

Narrowing is one of the primary methods for implementing functional logic programming languages. Property-based testing is an automatic approach to assuring the correctness of software systems. In recent years, a number of systems have been developed that seek to apply the benefits of narrowing in the area of property-based testing. This thesis considers two limitations with these systems. First of all, most of the existing narrowing-based testing tools have focused on practical issues and lack supporting theory. And secondly, these tools typically only perform well on properties that have particular forms. We address these limitations by developing an approach to narrowing that is both practical and principled and demonstrate how this can be used to expand the range of properties that can be automatically tested using a narrowing-based approach.

Formal Methods for Constraint-Based Testing and Reversible Debugging in Erlang

ADRIÁN PALACIOS
Universitat Politècnica de València, Spain

Date: February 2020; Advisor: Germán Vidal
URL: https://tinyurl.com/yagyqrnp

Erlang is a message-passing concurrent, functional programming language based on the actor model. These and other features make it especially appropriate for distributed, soft real-time applications. In the recent years, Erlang's popularity has increased due to the demand for concurrent services.

However, developing error-free systems in Erlang is quite a challenge. Although Erlang avoids many problems by design (e.g., deadlocks), some other problems may appear. Here, testing and debugging techniques based on formal methods may be helpful to detect, locate, and fix programming errors in Erlang.

In this thesis, we propose several methods for testing and debugging in Erlang. In particular, these methods are based on semantics models for concolic testing, property-based testing, causal-consistent reversible debugging, and causal-consistent replay debugging of Erlang programs. We formally prove the main properties of our proposals and design open-source tools that implement these methods.

Formalization of Transform Methods Using Higher-Order-Logic Theorem Proving

ADNAN RASHID
National University of Sciences & Technology, Pakistan

Date: March 2019; Advisor: Osman Hasan
URL: https://tinyurl.com/y86346fm

Transform methods, such as the Laplace and the Fourier transforms, are widely used for analyzing the differential equations modeling the continuous dynamics of the engineering and physical systems. Traditionally, the transform methods based analysis is performed using paper-and-pencil proof and computer-based simulation techniques, such as symbolic and numerical methods. However, due to their inherent limitations, such as the human-error proneness of paper-and-pencil proof methods and the presence of unverified symbolic algorithms, discretization, and numerical errors in the simulations methods, these techniques cannot provide accurate results. The incomplete and inaccurate analysis poses a serious threat to the safety-critical domain, such as medicine and transportation, of engineering systems. To overcome these limitations, we propose to use higher-order-logic theorem proving to reason about the continuous dynamics of the engineering and physical systems using transform methods. The main advantages of this approach are the high expressiveness of the higher-order logic and the soundness of theorem provers, which provide absolute accuracy of the analysis. In particular, this thesis presents a higher-order-logic formalization of the Laplace and the Fourier transforms, which includes their formal definitions and the formal verification of their classical properties, such as linearity, time shifting, frequency shifting, time scaling, integration, differentiation, and uniqueness. The formal reasoning about these properties involves multivariable calculus theories, that is, differential, integration, topological, and vectors theories. Based on the availability of these theories in the HOL Light theorem prover, we chose it for our work. This thesis also provides the formal verification of a relationship between various transform methods, that is, the relationship between the Laplace and the Fourier transforms, and the relationship between the Fourier transform and the Fourier Cosine and Sine transforms. The proposed formalization plays a vital role in formally verifying the solutions of differential equations in both the time and the frequency domain and thus facilitates formal dynamical analysis of these systems. To illustrate the practical utilization and effectiveness, we use our proposed formalization for formally analyzing a 4-π soft error crosstalk model for Integrated Circuits (ICs), an audio equalizer, an Unmanned Free swimming Submersible (UFSS) vehicle, and a platoon of automated vehicles using HOL Light.