JFP **33**, e15, 3 pages, 2024.
doi:10.1017/S0956796823000138

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, i.e. 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 two 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

Orchestrating the Internet of Things with Task-Oriented Programming — A Purely Functional Rhapsody

MART LUBBERS
Radboud University Nijmegen, The Netherlands

Date: October 2023; Advisor: Rinus Plasmeijer, Pieter Koopman and Jan-Martin Jansen
URL: https://tinyurl.com/436appc6

Developing reliable software for the Internet of Things (IoT) is difficult because IoT systems are dynamic, interactive, distributed, collaborative, multi-tiered, and multitasking. The complexity is increased further by semantic friction that arises through different hardware and software characteristics between tiers. Many computers that operate in IoT systems are edge devices that interact with the environment using sensors and actuators. Edge devices are often powered by low-cost microcontrollers designed for embedded applications. They have little memory, unhurried processors, and are slow in communication but are also small and energy efficient.

Task-oriented programming (TOP) can cope with the challenges of IoT programming. In TOP, the main building blocks are tasks, an abstract representation of work. During execution, the current value of the task is observable, and other tasks can act upon it. Collaboration patterns are modelled by combining and transforming tasks into compound tasks. Edge devices benefit from TOP as well, but running TOP systems within the limitations of resource-constrained microcontrollers is not straightforward.

This dissertation demonstrates how to include edge devices in TOP systems using DSLs. With these techniques, all tiers and their interoperation of an IoT system are specified in a single high-level source, language, paradigm, high abstraction level, and type system. First, I present advanced DSL embedding techniques. Then mTask is shown, a TOP DSL for IoT edge devices, embedded in iTask. Tasks are constructed and compiled at run time to allow tailoring tasks to the current work requirements. The task is then sent to the device for interpretation. A device is programmed once with a light-weight domain-specific OS. This OS executes tasks in an energy-efficient way and automates all communications and data sharing. All aspects of the mTask system are shown: example applications, language design, implementation details, integration with iTask, and green computing facilities.

Finally, tierless IoT programming is compared to traditional tiered programming. In tierless programming frameworks, the size of the code and the number of required programming languages is reduced significantly. By using a single paradigm and a system-wide type system, tierless programming reduces problems such as semantic friction; maintainability and robustness issues; and interoperation safety.

Programming with Monoidal Profunctors and Semiarrows

ALEXANDRE GARCIA DE OLIVEIRA
University of Sao Paulo, Brazil

Date: September 2023; Advisor: Mauro Javier Jaskelioff and Ana Cristina Vieira de Melo
URL: `https://tinyurl.com/227hnc93`

This work investigates monoidal profunctors and their extensions, such as effectful monoidal profunctors and semiarrows, as tools for reasoning and structuring pure functional programs from a categorical perspective and within a Haskell implementation. We approach them as monoids within a specific monoidal category of profunctors and as semiarrows in a semiarrow category. We examine the properties of this monoidal category and construct and implement the free monoidal profunctor. Furthermore, we detail the properties and laws of a semiarrow, deriving examples of its usage and highlighting its potential for effectively managing delays in synchronous programs. Moore machines serve as an illustrative example. Additional applications include optics and a monoidal profunctor structure-preserving connection between Moore machines, left scans, and left folds.