Computer Science Tripos - Part II - Project Proposal

E. While, Fitzwilliam College

October 2020

Introduction

There is a well documented correspondence between intuitonistic natural deduction and the Simply-Typed Lambda Calculus (STLC), but it was Griffin [2] who first introduced the idea that there is an equivalent correspondence between languages with control operators and the constructs of classical logic. Possibly the most well-known example of this relationship is Parigot's $\lambda\mu$-calculus [3] that he showed corresponded with classical natural deduction. Philip Wadler then expanded on this and presented what he called the 'Dual Calculus' [4] – corresponding to the classical sequent calculus, rather than natural deduction. In Wadler's system conjunction, disjunction, and negation are primitive and implication is defined in terms of those connectives, with different definitions for Call-by-Value (CBV) and Call-by-Name (CBN). He used this to give formulations of CBN and CBV, each with a separate set of reduction rules, that are clearly duals of each other. He then also goes onto provide continuation-passing style (CPS) translations from the Dual Calculus to the STLC, giving us the rather surprising result that the Dual Calculus is no more powerful, corresponding to the equally surprising result that classical logic is no more powerful than intuitionistic logic.

The aim of my project will be to formalise this Dual Calculus and the duality theorems, and then implement the CPS translations to the STLC in the programming language Agda. I will do this using the type– and scope–safe representation of higher-order languages in dependently-typed proof assistants, and will then use this formalisation to prove certain properties of the calculus, and of the CPS translation. Agda is a dependently typed functional programming language that is also a proof assistant based on the propositions-as-types paradigm.

Resources Required

The software that this project requires, such as Agda (plus its standard library) and agda-mode for VSCode, is open-source. I plan to use my own computer

for working on the project. Its specifications are AMD Ryzen 5 Pro 3500U 2.10 GHz, 16 GB RAM, 256 GB SSD, running Windows 10. I accept full responsibility for this machine, and I have made contingency plans to protect myself against hardware and/or software failure. My contingency plan will be to store all my code in a Git repository hosted on Github, and to also create a backup of my most recent code onto a memory stick twice a week. I will do all my write up in Overleaf so it will be stored remotely. Should my computer fail, I will be able to easily transfer my work to one of the MCS machines. I do not require any other special resources.

Starting Point

As far as I am aware there does not currently exist a formalisation of the dual calculus in Agda nor a machine-checked proof of its duality properties or relationship with the CPS.

To prepare for the project I have read Wadler's papers [4] [5] on the duality of CBV and CBN as well as Agestam's paper [1] on interpreting Classical Logic using the lambda calculus to familiarise myself with the wider subject area of languages corresponding to different forms of logic.

Since I had no experience with Agda or any other proof assistant before starting this project I started working through the textbook Programming Language Foundations in Agda (PLFA) by Wadler [6] in September when it became appartent I would be using it for my project.

Work to be done

I have broken down the work to be completed into the following key components:

1. Learning Agda to the standard necessary to define a formalisation of the dual calculus and write proofs for its properties.
2. Encoding the syntax of the dual calculus in Agda, using the type– and scope–safe representation of higher-order languages in dependently-typed proof assistants.
3. Defining the dual translation of the calculus to itself and proving its relevant properties (e.g. that duality is an involution and a term is only derivable iff its dual is derivable)
4. Implementing an interpreter of dual terms to Agda programs using the Call-by-Name and Call-by-Value CPS translation
5. Proving that the Call-by-Name and Call-by-Value translations are dual.
6. Encoding proof terms of classical theorems and other example programs in the dual calculus and comparing their execution after CBV and CBN translation

Success Criteria

These are the goals that can be used to judge whether or not the project was a success:

1. Be competent enough with Agda to complete the other goals of the project
2. Formalise the Dual Calculus in Agda
3. Define the dual translation of the calculus to itself.
4. Prove the dual translation is an involution and that a term is only derivable if its dual is derivable
5. Implement an interpreter of dual terms to Agda programs using the CBV and CBN CPS translations
6. Prove that the CBV and CBN translations are duals of each other

Possible Extensions

If I have the time there are few extra things it would be useful or interesting to do, they are:

1. Defining an equational theory or operational semantics for the dual calculus
2. Proving that the CPS translation into lambda terms is sound; it translates equal dual terms to equal Agda programs
3. Formalising the $\lambda\mu$-calculus and establishing its equational correspondence with the dual calculus

Timetable

- **26/10/20 – 08/11/20**

  Research into the Dual Calculus and related areas, research and practice writing effective Agda

  Milestone: Complete PLFA, be able to use Agda at required level, have good understanding of the project's theoretical background.

- **09/11/20 -– 22/11/20**

  Encoding the syntax of the dual calculus in Agda

  Milestone: Have a datatype that will only accept proper terms of the dual calculus

- **23/11/20 – 06/12/20**

  Define the dual translation of dual calculus terms

  Milestone: Have a working implementation to translate dual calculus terms to their duals

- **07/12/20 – 20/12/20**

  Prove the dual translation is an involution and that a term is only derivable iff its dual is derivable

  Milestone: Proof that dual translation is an involution is complete

  Milestone: Proof that a term is only derivable iff its dual is derivable is complete

- **21/12/20 – 12/01/21**

  Time off from project for a break over Christmas and Data Science coursework

- **11/01/21 – 17/01/21**

  Catch up if behind schedule, continue with future work if not

  Milestone: All previous milestones have been met

- **18/01/21 – 31/01/21**

  Implement the the interpreter of dual terms to Agda programs using the CBV and CBN CPS translation

  Milestone: Have a working function to translate dual calculus terms to equivalent CBV or CBN CPS Lambda Calculus terms

  Milestone: Progress report has been completed

- **01/02/21 – 14/02/21**

  Proving that CBV is the dual to CBN, work on extensions if there is time

  Milestone: Proof that CBV and CBN are duals of each other is complete.

- **15/02/21 – 28/02/21**

  Evaluate project by experimenting with example programs and proofs of classical theorems encoded in the dual calculus, and comparing their CPS translations under the two evaluation strategies

  Milestone: Have all the evaluation data that I require.

- **01/03/21 – 14/03/21**

  Time off from project for Interaction with ML coursework

- **15/03/21 – 28/03/21**

  Begin writing dissertation

  Milestone: Introduction, Preparation, and Implementation written and given to supervisor for comments.

- **29/03/21 – 11/04/21**

  Continue writing dissertation, incorporating comments on earlier chapters

  Milestone: First draft of dissertation has been completed and given to supervisor for comments

- **12/04/21 – 25/04/21**

  Finish dissertation write up, incorporating comments from first draft

  Milestone: Second draft of dissertation has been completed and given to supervisor for comments

  Milestone: Final draft of dissertation has been completed and submitted for grading alongside the code I have written.

- **26/04/21 – 14/05/21**

  Extra 2 and a half weeks until deadline to be safe, I will allocate these extra days to other pieces of work if needs be.

References

[1] Freddie Agestam. Interpretations of classical logic using λ-calculus.

[2] Tim Griffin. A formulae-as-types notion of control. In *Proceedings of POPL*, 1990.

[3] Michel Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. *Logic Programming and Automated Reasoning*, pages 190–201.

[4] Philip Wadler. Call-by-value is dual to call-by-name. In *Proceedings of ICFP'03*, Uppsala, Sweden, August 2003.

[5] Philip Wadler. *Call-by-Value is Dual to Call-by-Name – Reloaded.* 2005.

[6] Philip Wadler, Wen Kokke, and Jeremy G. Siek. *Programming Language Foundations in Agda.* July 2020. Available at http://plfa.inf.ed.ac.uk/20.07/.