**Introduction**

The correspondence between intuitionistic natural deduction and the simply-typed lambda calculus (STLC) is well-documented, slightly less well-known though is the equivalent correspondence between classical natural deduction and the lambda-mu calculus, this was introduced by Parigot building on the work of Griffin who introduced us to the correspondence between control operators in programming languages and constructs of classical logic. Phillip Wadler built on this foundation and presented what he called the ‘Dual Calculus’ – corresponding to the classical sequent calculus (rather than natural deduction), using this to give formulations of Call-by-Name (CBN) and Call-by-Value (CBV) which he then went on to show were de Morgan 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 by creating an intrinsically typed representation of the calculus, and then to use this formalisation to prove certain properties of the calculus, such as Type Preservation and Progress, 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 that I currently have no experience in.

**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, I will 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 know there does not currently exist a formalisation of the dual calculus nor of the properties of it that Wadler describes.

To prepare for the project I have read Wadler’s papers on the duality of CBV and CBN as well as Agestam’s paper 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.

**Work to be done**

Learning Agda to the standard necessary to define a formalisation of the dual calculus and write proofs for its properties. This will be done by continuing my progress through PLFA version 20.07 and completing the exercises within it.

Encoding the syntax of the dual calculus in Agda, this will be done by using an intrinsically typed representation of terms.

Defining the dual translation of terms in the dual calculus

Proving the dual translation is an involution and that a term is only derivable if its dual is derivable

Implementing the CPS translation from the dual calculus to the CBV and CBN CPS in the Lambda Calculus

Proving that CBV is the dual to CBN

**Possible Extensions**

Proving the translation from the dual calculus to the CPS STLC is sound

Defining an operational semantics of the dual calculus

Implementing a translator from the lambda-mu calculus to the dual calculus.

**Success Criteria**

Be competent enough with Agda to complete the other goals of the project

Formalise the Dual Calculus in Agda

Define the dual translation of terms in the dual calculus

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

Implement the CPS translation from the dual calculus to the CBV and CBN CPS in the Lambda Calculus

Prove that CBV is the dual of CBN

**Timetable**

Starting point is 26/10/2020

|  |
| --- |
| Block 1 (26/10 – 08/11)  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.* |
| Block 2 (09/11 – 22/11)  Encoding the syntax of the dual calculus in Agda  *Milestone: Have a datatype that will only accept proper terms of the dual calculus* |
| Block 3 (23/11 – 06/12)  Define the dual translation of dual calculus terms  *Milestone: Have a working implementation to translate dual calculus terms to their duals* |
| Block 4 (07/12 – 20/12)  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* |
| Block 5 (21/12 – 27/12)  Not project related – take some time off |
| Block 6 (28/12 – 10/01)  Not project related – Data Science Coursework |
| Block 6.5 (11/01 – 17/01)  Catch up if behind schedule, continue with future work if not  *Milestone: All previous milestones have been met* |
| Block 7 (18/01 – 31/01)  Implement the CPS translation from the dual calculus to the CBV and CBN CPS in the Lambda Calculus, and produce written progress report  *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* |
| Block 8 (01/02 – 14/02)  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.* |
| Block 9 (15/02 – 28/02)  Evaluate project  *Milestone: Have all the evaluation data that I require.* |
| Block 10 (01/03 – (14/03)  Not project related – Interaction with ML coursework |
| Block 11 (15/03 – (28/03)  Begin writing dissertation  *Milestone: Introduction, Preparation, and Implementation written and given to supervisor for comments.* |
| Block 12 (29/03 – 11/04)  Continue writing dissertation, incorporating comments on earlier chapters  *Milestone: First draft of dissertation has been completed and given to supervisor for comments* |
| Block 13 (12/04 – 25/04)  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.* |
| Block 14 (26/04 – 14/05)  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. |