Final Year Project Proposal

George Kaye
1522391
Supervisor: Noam Zeilberger

October 2018

1 Introduction

1.1 The λ-calculus

The λ-calculus is a model of computation that represents programs as function applications and variable abstraction. It is the underlying basis of all functional programming languages (such as OCaml and Haskell). The main method of computation in the λ-calculus is *β-reduction* – applying functions to their arguments using substitution:

$$(\lambda x.T)\, u \to_{\beta} T\, [x \mapsto u]$$

1.2 Rooted 3-valent maps

In graph theory, a map is a graph that has been *embedded* into a plane, creating *faces* in between the edges [1]. A *3-valent* map is a map where all of the vertices have a *valency* of three – i.e. three edges that connect to it. This map becomes *rooted* with the addition of a special node (the root) that has one edge connecting it to the map.

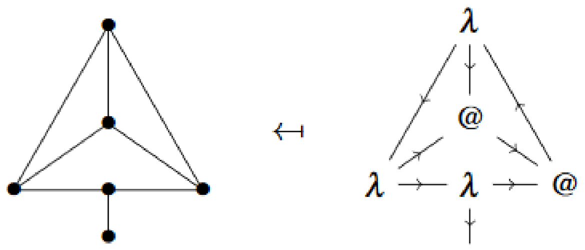

Figure 1: A representation of the term $\lambda x.\lambda y.\lambda z.x(yz)$ as a rooted 3-valent map, without and with node labels. From reference [2].

1.3 Representing λ-terms as maps

Linear λ-terms (in which each variable is only used once) can in fact be represented as these rooted 3-valent maps [2]. An example is shown in Figure 1. The idea of representing λ-terms as maps is not new – Wadsworth [3] and Statman [4] studied it for their PhD theses. More recent research has uncovered links between linear λ-calculus and the combinatorics of the rooted 3-valent maps [5].

2 Problem

Normalisation is the process of repeatedly applying β-reduction with the aim of reaching a normal form. While any λ-term has a normal form, it is not always possible to reach it – terms such as $\Omega = (\lambda x.xx)(\lambda x.xx)$ never terminate. Church showed that finding if a λ-term has a computable β-normal form is undecidable [6]. If we restrict ourselves to linear terms, the problem has been found to be decidable in polynomial time, and it is actually complete for the complexity class P [7].

However for other fragments of the λ-calculus such as *planar* terms (where variables are used in order – these terms can be represented by planar maps), the lower bound on the complexity of normalisation is unknown. The overall aim of this project would be to make some headway on this problem by developing a tool for generating terms from these fragments and visualising them as rooted 3-valent maps. We could then explore various properties of these λ-calculus fragments.

3 Timeline

3.1 A graphical representation of λ-terms (Weeks 1-5)

The first stage of the project will be to implement the λ-calculus (including elements such as α-equivalence and β-reduction) and develop a method of visualising λ-terms as rooted 3-valent maps, such as the earlier example in Figure 1. This can then be extended to include evaluation and normalisation.

3.1.1 Tasks

- Implement the basic elements of the λ-calculus in Javascript.
- Convert user input into de Bruijn notation.
- Implement β-reduction and evaluation.
- Draw λ-terms on the screen as rooted 3-valent maps.
- Implement normalisation.

3.2 Generating λ-terms from a given fragment (Weeks 6-11)

The next stage will be to develop a way of generating all the λ-terms of a given fragment (e.g. terms with two lambdas) and display them in a 'gallery' using the visualiser developed in part 1. This can also be used to enumerate the number of λ-terms of this fragment – the amount of terms in a given fragment can sometimes form interesting patterns.

For example, if we enumerate the closed linear λ-terms with two lambdas we find that there are five:

- $\lambda x.\lambda y.xy$
- $\lambda x.\lambda y.yx$
- $\lambda x.x\,\lambda y.y$
- $(\lambda x.x)(\lambda y.y)$

- $\lambda x.(\lambda y.y)\, x$

The generator would find these five terms and display them together, allowing us to compare and contrast each one. We can also represent these terms as *combinatorial maps* which can be used to compute interesting topological properties such as genus. An example of combinatorial maps can be found in [2].

3.2.1 Tasks

- Research ways of generating λ-terms.
- Implement a generator for various fragments of the λ-calculus.
- Show all (or a suitable number) of the λ-terms for a chosen fragment on screen.
- Represent these λ-term graphs as combinatorial maps.

3.3 Studying normalisation properties of fragments (Weeks 12-22)

Once we can define and generate fragments of the λ-calculus, we will be able to investigate properties of different fragments. One important property to examine is normalisation of terms. We can generate normalisation graphs (representing the various sequences of β-reductions that lead to a normal form). From this we can observe several things, such as path length or whether there are multiple different routes to take (an example of this can be seen in Figure 2). This ties in directly with the problem of normalisation complexity.

3.3.1 Tasks

- Generate normalisation graphs for λ-terms and display these.
- Explore these normalisation graphs.

$$\begin{tikzcd}
\lambda x.(\lambda y.\ y\ x)\ (\lambda z.\ z) \arrow[d] & & \lambda x.\ (\lambda y.\ y)\ ((\lambda z.\ z)\ x) \arrow[dl] \arrow[dr] & \\
\lambda x.\ (\lambda z.\ z)\ x \arrow[d] & \lambda x.\ ((\lambda z.\ z)\ x) \arrow[dr] & & \lambda x.\ (\lambda y.\ y)\ x \arrow[dl] \\
\lambda x.\ x & & \lambda x.\ x &
\end{tikzcd}$$

Figure 2: Examples of normalisation graphs. Arrows represent a β-reduction.

3.4 Studying other properties of fragments

If enough progress is made with regards to normalisation, it might be interesting to study some other properties of these λ-calculus fragments. For example, typing of the λ-terms in the maps could be studied, as this links to normalisation. Alternatively, topological properties of the generated maps, such as genus, diameter or connectedness could be investigated.

3.4.1 Tasks

- Explore typing of generated λ-terms.
- Explore topological properties of generated λ-terms.

4 Software

The main tool will be developed in Javascript. This means that it can be accessed via a website and will not have many dependencies for others to use it.

References

[1] S. K. Lando and A. K. Zvonkin (2004). *Graphs on Surfaces and Their Applications. Number 141 in Encyclopaedia of Mathematical Sciences. Springer.*

[2] N. Zeilberger (2016). *Linear lambda terms as invariants of rooted trivalent maps.*

[3] C. P. Wadsworth (1971). *Semantics and Pragmatics of the Lambda-Calculus. PhD thesis. Oxford University.*

[4] R. Statman (1974). *Structural complexity of proofs. PhD thesis.*

[5] O. Bodini, D. Gardy, and A. Jacquot (2013). *Asymptotics and random sampling for BCI and BCK lambda terms. Theoretical Computer Science 502, 227–238.*

[6] A. Church (1936). *An unsolvable problem of elementary number theory.*

[7] H. G. Mairson (2004). *Linear lambda calculus and PTIME-comp-leteness. J. Functional Programming 14, 6 (Nov. 2004), 623–633.*