Computation Theory

Exercise Sheet 3

Alistair O'Brien

February 3, 2024

Before Attempting the Problems

The primary goal of this course is to introduce to you *formal notion of computability* and various computational models, such as *register machines*, *Turing machines*, and the *lambda calculus*. It is vital that you master many of these topics for various other Tripos courses[1].

Please complete these exercises and submit your solutions **96 hours** before our scheduled supervisions to `ajo41@cam.ac.uk` as a **PDF** attachment. Feel free to choose the format of your answers (handwritten, typed, LaTeX'd) – just ensure that they're (mostly) legible[2].

Attempt to complete at least 75% of the set exercises – if you get stuck simply make a note of it in your answer and move onto the next question. We will be able to discuss all the solutions during the supervisions.

These supervisions will focus on both examinable and non-examinable material, the latter being for pedigogical reasons. Each week I will provide various *practical programming exercises* related to the course content. You are welcome to skip these questions, they are simply there for your benefit.

Some of the exercises are taken from Prof. Dawar's exercise sheets which you can find here: https://www.cl.cam.ac.uk/teaching/2324/CompTheory/exercise-sheet.pdf.

[1]In particular, Complexity Theory (IB), Semantics (IB), Denotational Semantics (II), Types (II), Hoare Logic and Model Checking (II), and Category Theory (II)

[2]Don't worry too much about this though, I can't handwrite anything legible either

Exercises

1. For each $n \in \mathbb{N}$, let g_n be the function mapping each $y \in \mathbb{N}$ to the value $ack(n, y)$ of Ackermann's function at $(n, y) \in \mathbb{N}^2$.
   (a) Show for all $(n, y) \in \mathbb{N}^2$ that $g_{n+1}(y) = (g_n)^{y+1}(1)$, where $h^{(k)}(z)$ is the result of k repeated applications of the function h to initial argument z.
   (b) Deduce that each g_n is a primitive recursive function.
   (c) Deduce that Ackermann's function is total recursive.

2. Give a recursive definition of the function $len(M)$ denoting the length of the λ-term $M \in \Lambda$ given by the total number of variables in M.

3. (a) Define the *subterm* relation $M \sqsubseteq N$ inductively. For example:

   $$x \sqsubseteq \lambda y.ux \qquad \lambda x.y \sqsubseteq \lambda x.y \qquad x\ y \sqsubseteq (\lambda x.x\ y) \qquad u\ v \not\sqsubseteq \lambda x.x\ u\ (v\ y)$$

   (b) We say that there is an *occurrence* of M in N if $M \sqsubseteq N$.
      i. Mark all occurrences of xy in $(xy)(\lambda x.xy)$.
      ii. Mark all occurrences of x in $(xy)(\lambda x.xy)$.
      iii. Mark all occurrences of xy in $\lambda xy.xy$.
      iv. Mark all occurrences of uv in $x(uv)(\lambda u.v(uv))uv$.
      v. Does $\lambda u.u$ occur in $\lambda u.uv$?

4. Let $M \in \Lambda$ be the λ-term $M \triangleq \lambda xy.x(\lambda z.zu)y$.
   (a) What is the β-normal form of the term $N \triangleq M(\lambda vw.v(wb))(\lambda xy.yaz)$?
   (b) Apply the simultaneous substitution $\sigma \triangleq [x/y, (\lambda xy.zy)/u]$ to M and N, and find the β-normal form of $N[\sigma]$.
   (c) We define η-equivalence as: $M =_\eta \lambda x.Mx$ for any λ-term M. Give a shorter and a longer term η-equivalent to M. What is the use of η-equivalence in functional programming?

5. Prove of the correctness of Church addition

   $$\mathbf{Plus}\ \underline{m}\ \underline{n} =_\beta \underline{m+n}$$

6. Show that the λ-term $\mathbf{Ack} \triangleq \lambda x.x\ T\ \mathbf{Succ}$, where $T \triangleq \lambda fy.yf(f\underline{1})$ represents Ackermann's function $ack : \mathbb{N}^2 \to \mathbb{N}$

7. Consider the following λ-terms:

   $$\mathbf{I} \triangleq \lambda x.x \qquad \mathbf{B} \triangleq \lambda gfx.fx\ \mathbf{I}(g(fx))$$

   (a) Show that $\underline{n}\ \mathbf{I} =_\beta \mathbf{I}$ for all $n \in \mathbb{N}$.
   (b) Assuming the fact about normal order reduction mentioned on slide 115, show that if partial functions $f, g : \mathbb{N} \rightharpoonup \mathbb{N}$ are represented by closed λ-terms F and G respectively, then their composition $f \circ g$ is represented by $B\ F\ G$.

8. https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2021p6q6.pdf

9. (*optional*) Write a lambda calculus interpreter in OCaml. Write a compiler from L2 (IB semantics) into the lambda calculus.