15-354: CDM	K. Sutner
Assignment 2	Due: **Sep. 13, 2024**.

1. Loopy Loops (40)

Background

Consider a small programming language LOOP that has only one data type, natural numbers. The syntax is described in the following table:

constant	$0 \in \mathbb{N}$
variables	x, y, z, ... ranging over $\mathbb{N}$
operations	increment $x{+}{+}$
assignments	$x = 0$ and $x = y$
sequential composition	$P; Q$
control	`do` $x : P$ `od`

The semantics are obvious, except for the loop construct: `do` $x : P$ `od` is intended to mean: "Let n be the value of x before the loop is entered; then execute P exactly n times." Thus, the loop terminates after n rounds even if P changes the value of x. For example, the following LOOP program computes addition:

```
//  add : x, y --> z
        z = x;
        do y :
              z++;
        od
```

Here x and y are input variables, and the result is in z. We assume that all non-input variables are initialized to 0. So, we have a notion of a LOOP-computable function (this is entirely analogous to our definitions for register machines).

Task

A. Show how to implement multiplication and the predecessor function as LOOP programs.

B. What function does the following loop program compute?

```
// mystery : x --> x
   do x:
      do x: x++ od
   od
```

C. Show that every primitive recursive function is LOOP-computable.

D. Show that every LOOP-computable function is primitive recursive.

E. Informally, what is the key difference between LOOP and register machine programs?

2. Register Machines and Sequence Numbers (30)

Background

Recall the coding function for sequences of natural numbers introduced in class:

$$\pi(x, y) = 2^x(2y + 1)$$
$$\langle \mathsf{nil} \rangle = 0$$
$$\langle a_1, \ldots, a_n \rangle = \pi(a_1, \langle a_2, \ldots, a_n \rangle)$$

Task

A. Give a simple bound on $\langle a_1, \ldots, a_n \rangle$ in terms of n and $\max a_i$.

B. Construct a register machine program `digcnt` that, on input x, returns the number of binary digits of x (no leading zeros).

C. Construct a register machine program `append` that, on input $\langle a_1, \ldots, a_n \rangle$ and b, returns $\langle a_1, \ldots, a_n, b \rangle$.

D. Roughly, what is the running time of your programs?

Comment

Make sure to give a detailed explanation of how your programs work, plain RMP code drives the TA nuts. A flowgraph might be a good idea, too.

For the running time do not try to come up with a precise answer, just order of magnitude.

3. The Busy Beaver Function (RM) (30)

Background

The Busy Beaver function β is a famous example of a function that is just barely non-computable. For our purposes, let's define $\beta(n)$ as follows. Consider all register machines P with n instructions and no input (so all registers are initially 0). Executing such a machine will either produce a diverging computation or some output x_P in register R_0. Define $\beta(n)$ to be the maximum of all x_P as P ranges over n-instruction programs that converge.

It is intuitively clear that β is not computable: we have no way of eliminating the non-halting programs from the competition. Alas, it's not so easy to come up with a clean proof. One line of reasoning is somewhat similar to the argument that shows that the Ackermann function is not primitive recursive: one shows that β grows faster than any computable function.

Task

A. Show that, for any natural number m, there is a register machine without input that outputs m and uses only $O(\log m)$ instructions.

B. Assume $f : \mathbb{N} \to \mathbb{N}$ is a strictly increasing computable function. Show that for some sufficiently large x we must have $f(x) < \beta(x)$.

C. Conclude that β is not computable.

D. Prof. Dr. Blasius Wurzelbrunft sells a device called HaltingBlackBox™ that allegedly solves the Halting Problem for register machines. Explain how Wurzelbrunft's gizmo could be used to compute β.

Comment

The bound in part (A) is far from tight in special cases: some numbers m have much shorter programs: think about 2^{2^k}. But, in general $\log m$ is impossible to beat (Kolmogorov-Chaitin program-size complexity). Part (D) says that β is K-computable.