15-354: CDM **K. Sutner**

Assignment 2 Due: **Feb. 19, 2021**.

1. Loopy Loops (40)

Background

Consider a small programming language LOOP with the following syntax:

constants	$0 \in \mathbb{N}$
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.

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.

2. 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.

3. Binary Register Machines (30)

Background

A standard register machine operates on registers containing natural numbers, the only operations are increment and guarded decrement. Here is an alternative version of these machines that operate directly on bit strings, which we can think of as binary expansion of natural numbers. Thus register contains a sequence of bits

$$R_r : b_0, b_1, \ldots, b_n$$

where $n \geq 0$ may be arbitrarily large. Let's say we use LSD first, so the numerical value of the register contents is $[R_r] = \sum_{i \leq n} b_i 2^i$. Registers (other than input) are initialized to the single bit 0. A binary register machine (BRM) has the following instruction set:

- `zero r k l`
  Check if $[R_r] = 0$; if so, goto instruction k, otherwise goto instruction l.
- `lshft r k`
  Shift the contents of R_r to the left, goto instruction k.
- `rshft r k`
  Shift the contents of R_r to the right (padding with 0), goto instruction k.
- `set r k`
  Set the first bit of R_r to 1, goto instruction k.
- `read r s k`
  Read the first bit of R_r and write it into R_s, goto instruction k.
  Thus $[R_s] = 0$ or $[R_s] = 1$ depending on the first bit in R_r.

Task

A. Explain how to simulate a BRM on an ordinary RM. Make sure to explain the general strategy of your simulation.

B. Explain how to implement a BRM instruction `unset r k` which sets the first bit of R_r to 0 (and continues at instruction k).

C. Explain how to implement an increment operation `inc r k` on a BRM.

D. Explain how to implement a conditional decrement operation `dec r k l` on a BRM.

E. Show how to compute our pairing function $\pi(x, y) = 2^x(2y + 1)$ on a BRM.

F. Conclude that BRMs are computationally universal: argue that one can construct a universal BRM that can compute all arithmetic computable functions (say, of one argument).

Comment

Keep the answer to the last item short, there is no need to spell out all the gory details.