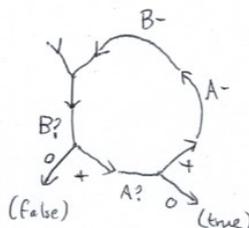


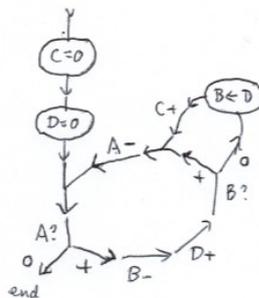
Informatics 2 – Introduction to Algorithms and Data Structures

Tutorial 10: Register machines and computability (SOLUTIONS)

1. (a) *Design a flowchart for a register machine that tests whether ‘ $A < B$ ’.*



- (b) *Design a machine that computes ‘ $A \text{ div } B$ ’ and ‘ $A \text{ mod } B$ ’ (assuming B is non-zero), storing the results in C and D respectively.*



Here, for clarity, we have assumed given some very simple components. ‘ $C=0$ ’ and ‘ $D=0$ ’ do what they say; ‘ $B \leftarrow D$ ’ copies the value of D to B , setting D to 0 in the process.

- (c) *Show that if both $f : \mathbb{N} \rightarrow \mathbb{N}$ and $g : \mathbb{N} \rightarrow \mathbb{N}$ are RM-computable, then so is their composition h defined by $h(n) = g(f(n))$.*

Saying f is RM-computable means that there’s a register machine F such that, for any $n \in \mathbb{N}$, if F is run on an initial state with $A = n$, it will terminate in a final state with $A = f(n)$. Likewise, g is RM-computable if there’s a machine G doing the same job for G . Given such machines, we may simply plug them together by connecting the exit point of F to the entry point of G . (Strictly

speaking, we first need to ensure F and G have the same number of registers, which we may do by adding extra (unused) registers to F or G as required.) the resulting machine will compute the composition h as required.

- (d) Show that if $e, f, g : \mathbb{N} \rightarrow \mathbb{N}$ are all RM-computable, then so is the function k defined by

$$k(n) = \text{if } e(n) = 0 \text{ then } f(n) \text{ else } g(n)$$

Suppose e, f, g are computed by register machines E, F, G respectively. Let r be two more than the maximum number of registers of E, F, G , and expand E, F, G to equivalent machines E', F', G' with r registers.

Our machine for computing k will work as follows, given an initial state with $A = n$.

- Copy n from A into the two spare registers, then copy one of them back to A .
 - Use E' to compute $e(n)$ (in A), then use 'A?' to branch on whether $e(n) = 0$.
 - On the 0 branch, copy the value of n back into A , then apply F' .
 - On the + branch, copy n back into A and apply G' .
 - Merge the two exit points into one.
2. (a) What about the predicate 'the machine coded by m , when applied to the inputs coded by n , halts within k steps'? Would you expect this to be RM-decidable? Informally justify your answer.

This is certainly decidable. Given m, n and k , it is a purely mechanical task to simulate the execution of machine m on input n for up to k steps. This simulation will complete within finite time, and by then we'll know if the computation in question halts within k steps.

So the given predicate is decidable by a mechanical procedure. By an informal appeal to Church's thesis, then, we expect it to be decidable by a register machine. (Alternatively, one could explicitly construct such a machine and show it did this, but life is too short.)

- (b) Let T be the set of all codes for register machines that compute some total function $\mathbb{N} \rightarrow \mathbb{N}$. It would be nice if there were some register machine that could tell us, given any $m, m' \in T$, whether the machines represented by m and m' gave rise to the same total function. Show however that no such machine is possible.

Suppose such a machine existed. Here's how we could use it to solve the halting problem.

- Given any m (coding a register machine) and n (coding a memory state), we can use our solution to (a) to construct a machine $P_{m,n}$ that computes the function

$$k \mapsto (1 \text{ if machine } m \text{ on input } n \text{ halts within } \leq k \text{ steps, } 0 \text{ otherwise})$$

- This machine $P_{m,n}$ will have a certain numerical code $p_{m,n}$. What's more, since the construction of $P_{m,n}$ is uniform in m and n , it will be possible to compute $p_{m,n}$ given m and n .
- The trick is to note that:
 - for any m, n , $p_{m,n}$ is the code for a machine computing a total function, i.e. $p_{m,n} \in T$,

- this total function will be the constant zero function $\lambda k.0$ if and only if ' m applied to n ' never halts.

So if we had a machine D that tested whether two codes in T yield the same total function, So we could solve the halting problem as follows: Let q be a fixed code for some simple machine that computes $\lambda k.0$. Now given m, n , compute $p_{m,n}$ as above, then use D to test whether $p_{m,n}$ and q represent the same total function.