CST Part IB *Computation Theory*
List of corrections to the 2019/20 lecture notes

30 January 2020

Pages 25 and 49: The universal machine on page 49 does not simulate register machines with erroneous halts properly, according to the current definition of "erroneous halt" on page 25. Instead, when it processes an instruction that contains jumps to non-existent labels, it carries out the instruction's register operation (increment of decrement-if-zero), sets the program counter to the appropriate non-existent label number and then halts (as described by the high-level specification on page 48).

Although the design of the machine on page 49 could be corrected, it is simpler to change the convention for erroneous halting to be as implemented by the current machine. A corrected page 25 is attached.

Register machine computation

A computation of a RM is a (finite or infinite) sequence of configurations

$$c_0, c_1, c_2, \ldots$$

where

- $c_0 = (0, r_0, \ldots, r_n)$ is an initial configuration
- each $c = (\ell, r_0, \ldots, r_n)$ in the sequence determines the next configuration in the sequence (if any) by carrying out the program instruction labelled $\mathtt{L}_\ell$ with registers containing $r_0, \ldots, r_n$.

Halting

For a finite computation $c_0, c_1, \ldots, c_m$, the last configuration $c_m = (\ell, r, \ldots)$ must be a halting configuration, i.e. ℓ satisfies:

either $\ell \geq$ number of instructions in program, so that there is no instruction labelled $\mathtt{L}_\ell$ (an "erroneous halt")

or ℓ^{th} instruction in program has body `HALT` (a "proper halt")

N.B. can always modify programs (without affecting their computations) to turn all erroneous halts into proper halts by adding extra `HALT` instructions to the list with appropriate labels.