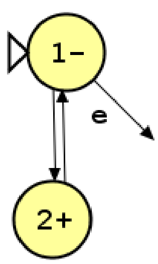
Computability and Logic

HW 7

Due: Friday, April 17

1. Use the Abacus machine software to create an Abacus machine that computes max(x,y).
2. Use the Abacus machine software to create an Abacus machine that computes the quo(x,y) and rem(x,y) functions at the same time by leaving quo in register 3, and rem in register 4 (make sure to leave x in register 1, y in register 2 when the machine halts, and empty out all registers beyond 4). For the special case where y = 0 we set quo(x,0) = 0 and rem(x,0) = x.
3. Questions A, B, C from the Abacus-Machine Halting Function handout.
4. Use the Turing-machine software to create a Turing machine that simulates the following Abacus machine:



Your Turing-machine should simulate this Abacus-machine using the following simulation conventions:

- Assume that the Abacus-machine has more than 2 registers (i.e. act as if this Abacus-machine is part of a larger machine)
- The Turing-tape is used to represent the contents of the registers using unary representation. To be specific, if the Abacus-machine registers R_1, R_2, … R_k contain the numbers n_1, n_2, … n_k respectively, then this will be represented using Turing-Tape configuration [n_1, n_2, … n_k] (as defined in HW 6)
- The Turing-machine starts at the leftmost 1 on the tape, and after each instruction of the Abacus-machine that the Turing-machine simulates, the Turing-machine should return the head back to the leftmost one.

So I am *not* looking for a Turing-machine that accomplishes whatever this Abacus-machine accomplishes merely in terms input and output behavior, but for a Turing-machine that does it using the above conventions!

For all Abacus-machines you create for the HW, follow the following guidelines:

- Use the Abacus machine software AMS 2.1 to create your machine and submit electronically.
- Unless stated otherwise, when using Abacus-machines to compute functions over natural numbers, use the ‘standard’ convention of changing input configuration [n_1, n_2, … n_k] into output configuration [n_1, n_2, … n_k, f(n_1, n_2, … n_k)] where [n_1, n_2, … n_k] represents a tuple of numbers <n_1, n_2, … n_k> as follows: for all $1 \leq i \leq n$: register R_i contains n_i, and all other registers are empty. In other words: make sure that when the machine is done, you have the input of k numbers back in the first k registers, the function value is in register k+1, and all other registers are empty.
- Organize the nodes and transitions so that your machine looks nice and is ‘readable’ (i.e. don’t leave your machine a spaghetti of connections). Try to keep crossing connections at a minimum.