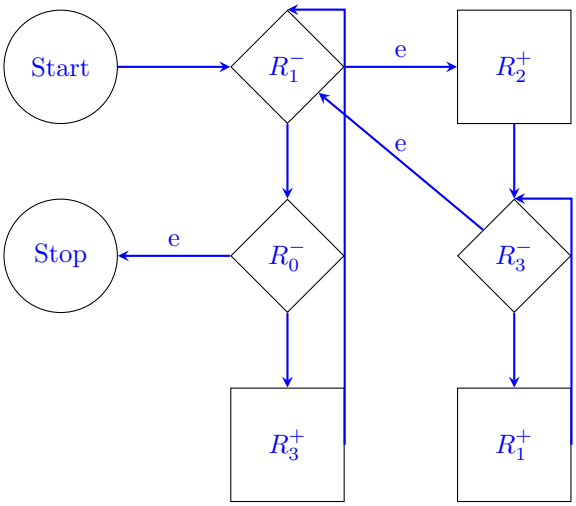
Math 29: More Register Machines

April 4th, 2022

1 More Computable Functions

Lemma 1. *Division is computable.*

Proof: Consider the following register machine program:



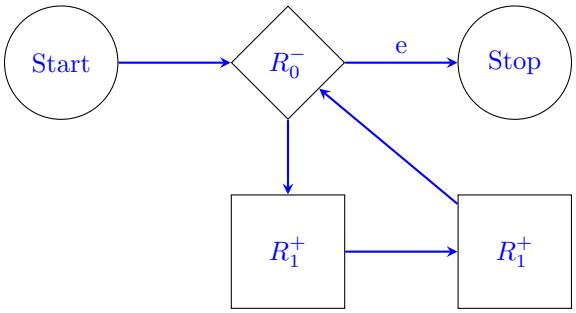
Upon entering the program, R_0 contains x and R_1 contains y. We are going to end up with $\frac{x}{y}$ in R_2. (This is integer division, which rounds down if x is not a multiple of y.)

Upon entering the program, we subtract y from x while simultaneously moving y into R_3. If at any point R_0 is empty, that means we end the computation. Once R_1 is empty, we add 1 to R_2 to keep track of how many times we have subtracted y from x. We then move y back into R_1 to repeat the first loop. When the program stops, R_2 will contain the number of times the first loop completed, i.e. how many times y goes into x. (Notice that the program loops infinitely if y is zero.)

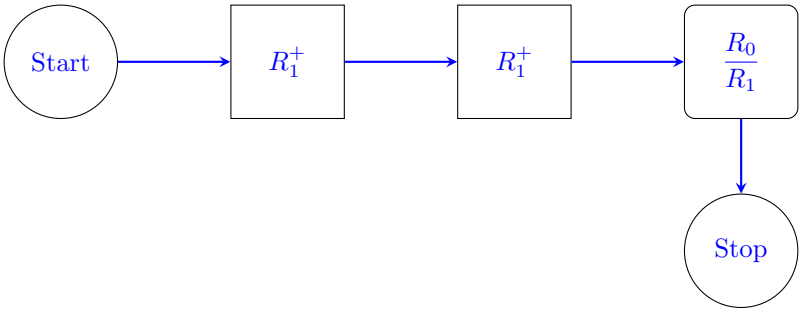
A set $X \subseteq \omega$ is computable if either its characteristic function or its principle function is computable.

Lemma 2. *The set of even numbers is computable.*

Proof: First, notice that the principal function of the set of even numbers is $p_E(n) = 2n$. Therefore the following register machine computes the principal function:



Alternately, we can compute the characteristic function with the help of the division register machine we built for the previous lemma.



We place 2 in R_1 then copy the division machine to divide R_0 by R_1. Reviewing the operation of our division machine, if R_0 was odd, then the R_0^- node will follow the empty node immediately after R_1^- activates the second time in the first loop, so R_1 contains a 0, which is the correct value for $\chi_E(n)$. If R_0 was even, then the R_0^- node will follow the empty node on the first run after the first loop resets, so R_1 will contain a 1 as desired.

Recall that, under our coding of the integers, the even numbers were the codes for the non-negative integers. So, the previous lemma not only told us that the set of even numbers is computable, but also that the the set of non-negative integers is computable.

Similarly, recall that under our canonical codes for finite sets of natural numbers, D_n is the set of natural numbers corresponding to the binary expansion of n. (In other words, the binary expansion of n corresponds to the characteristic function of D_n.) Then the even numbers are the ones whose first bit in the binary expansion is 0. This means that the even numbers correspond to the finite sets not containing 0. Therefore, the set of finite sets not containing 0 is computable!

Lemma 3. *The factorial function $n!$, where $0! = 1$ and $n! = n * (n-1)!$, is computable.*

Proof. Homework 2 Question 1. □

Lemma 4. *The Fibonacci function, where $f(0) = f(1) = 1$ and $f(n) = f(n-1) + f(n-2)$, is computable.*

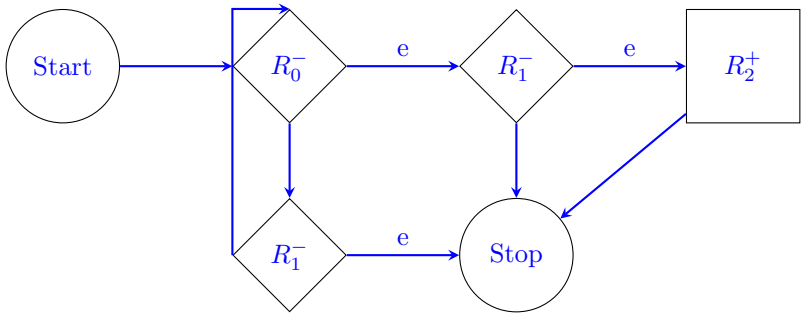
Proof. Homework 2 Question 2. □

A k-ary **relation** is a $\{0,1\}$-valued function whose domain consists of k-tuples. We think of 1 as representing "truth," and 0 as representing "false." For example, the equality relation $x = y$ is a 2-ary (or binary) relation which is 1 if x and y are the same number and 0 otherwise. We say a relation is computable if this function is computable. Equivalently, we can think of a k-ary relation as some set of k-tuples, with those tuples that satisfy the relation being in the set and those that do not being out of the set. The function above is the characteristic function of this set, so it is also equivalent to say a relation is computable if this set is computable.

For now, it is easiest to think of relations in the former terms to avoid needing to code and decode tuples in our register machines. This is because, for example, the characteristic function for the set of pairs (x, y) such that $x < y$ is technically different than the characteristic function for the set of codes of pairs $\langle x, y \rangle$ such that $x < y$. However, we will see later that these coding issues are a non-factor, at which point it will usually be simpler to think of relations in terms of sets rather than functions.

Lemma 5. *The equality relation is computable.*

Proof: We need to build a register machine such which computes the function $= (x, y)$ such that $= (x, y) = 1$ end up in R_2 if $x = y$ and $= (x, y) = 0$ ends up in R_2 otherwise. Consider the following:

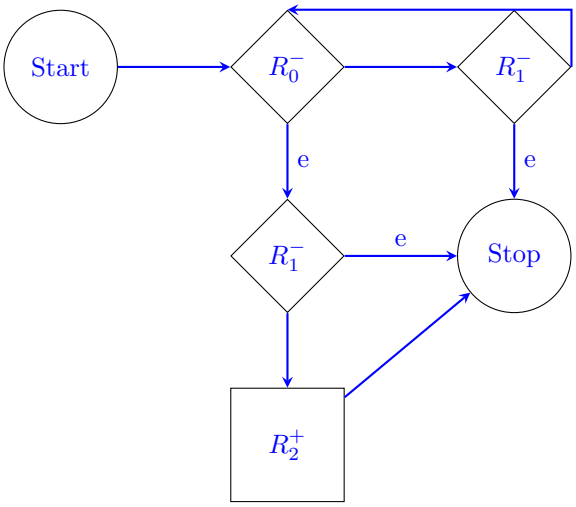


Note that if R_0 and R_1 contain the sme number, then the first loop is broken when R_0 is empty, at which time R_1 is empty as well, since the same number was subtracted from both. Therefore the second R_1^- node sends us to R_2^+, which then stops the program with the correct value in R_2. If the number in R_0 is larger than the number in R_1, then the first R_1^- will stop the program before we ever follow the empty node for the R_0^- node. If the number in R_1 is larger, then once the first loop breaks, R_1 will sitll not be empty, so the program will stop following the non-empty output node for the second R_1^- node.

Thus this register machine computes the equality relation as desired.

Lemma 6. *The less than relation is computable.*

Proof: Consider the following register machine:



We begin by removing one from R_0 and R_1 alternately. The number starting in R_0 is less than the number starting in R_1 exactly when R_0 is empty but R_1 is not, in which case we add one to R_2. In all other cases, we simply stop the program with the default value of 0 in R_2.