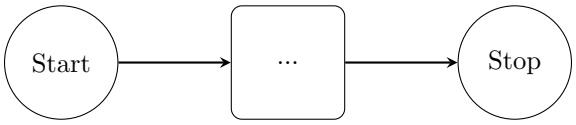
# Math 29: Register Machines

April 1st, 2022

## 1 A Model of Computing

Last time, we talked about coding objects as natural numbers, various properties of functions, and how we could think of sets as functions. Now, we'll start to develop the theory of what it means for a function from the natural numbers to the natural numbers to be (partially) computable, which will simultaneously develop the theory of computation for sets and objects which we can code using natural numbers.

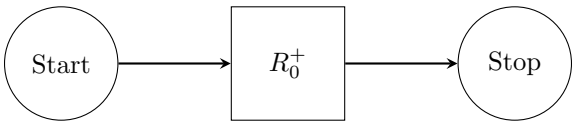
A **register machine** is represented by a flow-chart with a unique Start node, a unique Stop node, and finitely many nodes connected in-between. Program flow begins at the Start node and advances to the unique output node.



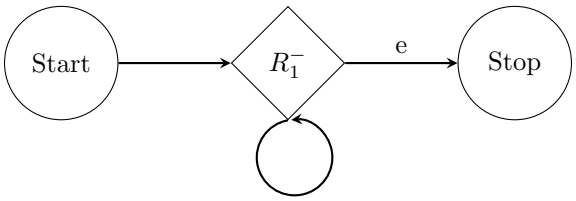

There are infinitely many **registers**, one for each natural number, which each contain a single natural number. Think of each register as a bucket which contains some number of balls. Between the start and stop nodes are some combination of **addition** nodes and **subtraction** nodes.


$i$ is the index of the register that the instruction is modifying. $+$ represents an addition node, and $-$ represents a subtraction node. The former puts one ball in one bucket, and the latter removes one ball from a register if there is one to be removed. In other words, the former adds one to the number in one register, and the latter subtracts one if the number in the register is not zero.

An addition node $R_i^+$ has one at least one input node and one output node. Once the program flow exits one of the input nodes, it enters the addition node. It then adds one to the value in the $i$-th register, and advances program flow to the output node. For example, the following simple program adds one to the first register then stops.



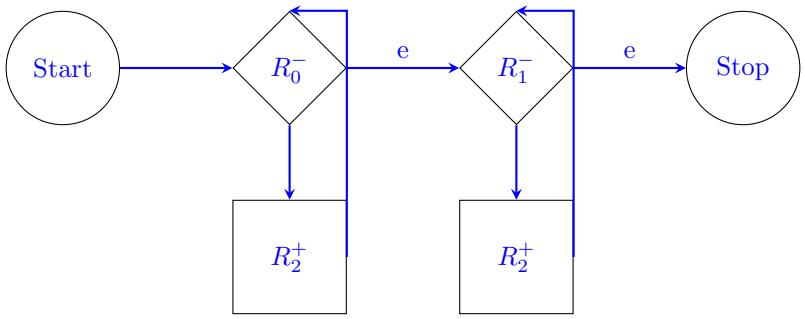

A subtraction node $R_i^-$ has at least one input node and two output nodes, one labeled $e$. Once program flow exits one of the input nodes, it enters the subtraction node. If the $i$-th register is empty, i.e. contains a 0, then it advances program flow to the output node labeled $e$. If the $i$-th register is not empty, then it is decremented by one and program flow advances to the other output node. For example, the following simple program empties the second register then stops.




Given a register machine $M$ and a natural number $k$, we get a partial function $M : \omega^k \to \omega$ such that $M(x_0, \ldots, x_{k-1})$ is the value in the register $R_k$ if the register machine runs with each $x_i$ in the $i$-th register, and 0 in all the other registers, and reaches the Stop node. (If we never reach the Stop node with the given inputs, then $M(x_0, \ldots, x_{k-1}) \uparrow$.) A function $f : \omega^k \to \omega$ is said to be **(register) computable** if there is some register machine $M$ such that $f(x_0, \ldots, x_{k-1}) = M(x_0, \ldots, x_{k-1})$ for all $x_0, \ldots, x_{k-1} \in \omega^k$. In other words, the register machine computes the value of $f$ on all inputs.

**Lemma 1.** *Addition, i.e. the 2-variable function* $+ : \omega^2 \to \omega$ *such that* $+(x, y) = x + y$*, is computable.*

**Proof:** We can verify that the following register machine program computes the addition function, where it starts with $x$ in $R_0$, $y$ in $R_1$, and 0's in all other registers:




Upon entering the first subtraction node, it removes one from the first register and adds one to the third, repeating until the first register is empty. The same then occurs for the second register. Once the second register is empty, we reach the Stop node. At this point, the third register started with a 0, and had a 1 added $x$ times in the first loop and $y$ times in the second loop, so it now containes $x + y$.

**Lemma 2.** *Multiplication, i.e. the 2-variable function* $* : \omega^2 \to \omega$ *such that* $*(x, y) = xy$*, is computable.*

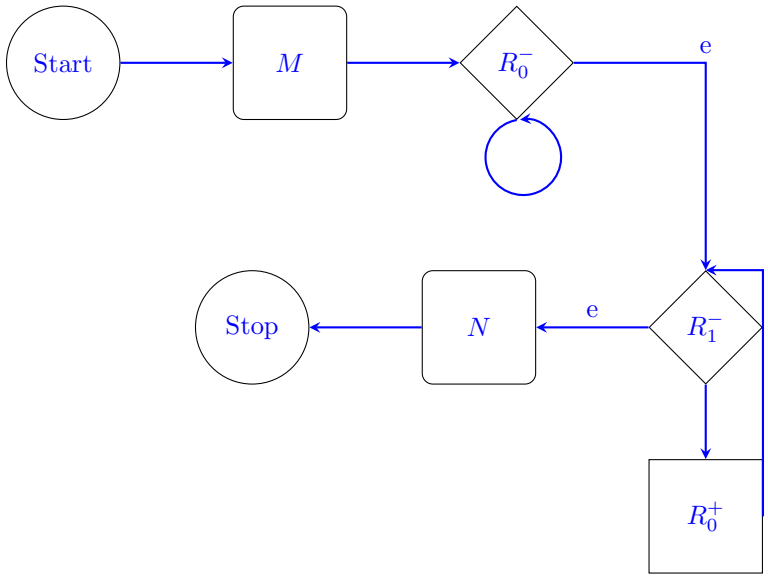
*Proof.* Homework 1: Question 5. □

**Lemma 3.** *Exponentiation, i.e. the 2-variable function* $e : \omega^2 \to \omega$ *such that* $e(x, y) = x^y$ *is computable.*

*Proof.* Homework 1: Question 6. □

**Lemma 4.** *If* $f : \omega \to \omega$ *and* $g : \omega \to \omega$ *are both computable, then their composition* $g \circ f$ *is also computable.*

**Proof:** As $f$ and $g$ are both computable, there are register machines $M$ and $N$ which, when $n$ is in $R_0$, output $f(n)$ and $g(n)$ in $R_1$ respectively. Therefore, consider the register machine




Here, the $M$ and $N$ blocks represent programming said register machines into the above outline, with the input arrows representing their start nodes and and the output arrows advancing from their stop nodes.

This program begins with $n$ in $R_0$ and runs the register machine $M$ to calculate $f(n)$, which ends up in $R_1$. Next, we empty $R_0$ and then move $f(n)$ into $R_0$. Then we run the the register machine $N$ to calculate $g(f(n))$, which is placed in $R_1$. Finally, we terminate the program.