80-311 Assignment 06

Karan Sikka
ksikka@cmu.edu
April 22, 2014

1.1

Addition can be recursively defined as follows:
$x + 0 = U_1^1(x)$
$x + (y + 1) = S(x + y)$.

1.2

The principle of induction is that if you have a well ordered set and you show that a statement is true for the least element of the set, and you show that "if the statement holds for an element then it holds for the successor of the element" then the statement holds for all elements in the set.

The statement we aim to prove is "$x + y = y + x$" for all x and y in the natural numbers.

Since equality works in both directions, we may fix y to be some arbitrary natural number without loss of generality. This allows us to prove the statement we aim to prove for all x and for some y, and the proven statement implies that $y + x$ is equal to $x + y$ for all y and some x where y, the first variable in this statement is fixed.

So we proceed by induction on x, assuming that y is arbitrary some natural number.

The base case is true, since $x + y = 0 + y = y = y + 0 = y + x$. Now we assume that $x + y = y + x$ for some x and y in the natural numbers. We want to show that $S(x) + y = y + S(x)$. By the associativity of the successor function, we see that We observe that $S(x) + y = S(x + y)$ (A TA said we could assume this in office hours).

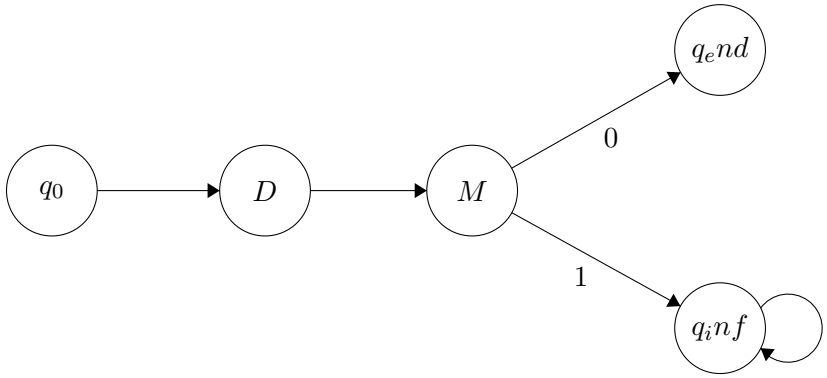
Then we can solve the inductive step by the following line of logic: $S(x) + y$
$= S(x + y)$
$= S(y + x)$ (by the induction hypothesis)
$= y + S(x)$

And thus we have proved the commutativity of addition by induction.

2

We describe a turing machine for N. We start on the start state q_0 and input i expressed on the tape. Then we transition to the start state of machine D from homework 9, and when the computation ends, the tape expresses (i, i). We transition from the end state of program D to the start state of program M. At the end state of program M, if the head is on a 0, we transition to an end state q_{end}. Else we self loop, over and over again, never halting.

Diagram:



3

Church's thesis states that all effectively calculable functions are computable and vice versa.

The backwards direction is trivial since a computable function is effectively calculable by the fact that there exists a turing mahine to carry out the computation.

The open questions is whether or not there exist functions which are effectivly calculable but not computable. The book makes the intuitive argument that if the function is calculable, it is presumabily so via a finite number of atomic steps, for which you could either define priminitive recursively or create a turing machine to perform.

It is hard to imagine a calculable function that is not computable. One could conceive that there are theoretical machine models which are more "powerful" than the turing machine. However it's been shown time and time again that those seemingly more powerful machines (TM with multidimensional tape) can actually be mapped to plain turing machines.

In many years of history, no one has produced a calculable function that is not computable.