The next theorem is known as the “s-m-n theorem,” for a reason that will be clear in a moment. The hard part is understanding just what the theorem says; once you understand the statement, it will seem fairly obvious.

**Theorem thy.1.** For each pair of natural numbers $n$ and $m$, there is a primitive recursive function $s^m_n$ such that for every sequence $x, a_0, \ldots, a_{m-1}, y_0, \ldots, y_{n-1}$, we have

$$\varphi^n_{s^m_n(x, a_0, \ldots, a_{m-1})}(y_0, \ldots, y_{n-1}) \simeq \varphi^{m+n}_x(a_0, \ldots, a_{m-1}, y_0, \ldots, y_{n-1}).$$

It is helpful to think of $s^m_n$ as acting on *programs*. That is, $s^m_n$ takes a program, $x$, for an $(m + n)$-ary function, as well as fixed inputs $a_0, \ldots, a_{m-1}$; and it returns a program, $s^m_n(x, a_0, \ldots, a_{m-1})$, for the $n$-ary function of the remaining arguments. It you think of $x$ as the description of a Turing machine, then $s^m_n(x, a_0, \ldots, a_{m-1})$ is the Turing machine that, on input $y_0, \ldots, y_{n-1}$, prepends $a_0, \ldots, a_{m-1}$ to the input string, and runs $x$. Each $s^m_n$ is then just a primitive recursive function that finds a code for the appropriate Turing machine.