

thy.1 The s - m - n Theorem

[cmp:thy:smn:sec](#) The next theorem is known as the “ s - m - n theorem,” for a reason that will [explanation](#) 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.

[cmp:thy:smn:thm:s-m-n](#) **Theorem thy.1.** *For each pair of natural numbers n and m , there is a primitive recursive function s_n^m such that for every sequence $x, a_0, \dots, a_{m-1}, y_0, \dots, y_{n-1}$, we have*

$$\varphi_{s_n^m(x, a_0, \dots, a_{m-1})}^n(y_0, \dots, y_{n-1}) \simeq \varphi_x^{m+n}(a_0, \dots, a_{m-1}, y_0, \dots, y_{n-1}).$$

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

Photo Credits

Bibliography