

## lam.1 Lambda Representable Functions are Computable

cmp:lam:cmp:  
sec

cmp:lam:cmp:  
thm:lambda-computable

**Theorem lam.1.** *If a partial function  $f$  is represented by a lambda term, it is computable.*

*Proof.* Suppose a function  $f$ , is represented by a lambda term  $X$ . Let us describe an informal procedure to compute  $f$ . On input  $m_0, \dots, m_{n-1}$ , write down the term  $X\overline{m_0} \dots \overline{m_{n-1}}$ . Build a tree, first writing down all the one-step reductions of the original term; below that, write all the one-step reductions of those (i.e., the two-step reductions of the original term); and keep going. If you ever reach a numeral, return that as the answer; otherwise, the function is undefined.

An appeal to Church's thesis tells us that this function is computable. A better way to prove the theorem would be to give a recursive description of this search procedure. For example, one could define a sequence primitive recursive functions and relations, "IsASubterm," "Substitute," "ReducesToInOneStep," "ReductionSequence," "Numeral," etc. The partial recursive procedure for computing  $f(m_0, \dots, m_{n-1})$  is then to search for a sequence of one-step reductions starting with  $X\overline{m_0} \dots \overline{m_{n-1}}$  and ending with a numeral, and return the number corresponding to that numeral. The details are long and tedious but otherwise routine.  $\square$

### Photo Credits

### Bibliography