rep.1 Currying

lam:rep:cur:sec

A λ-abstract $\lambda x.\, M$ represents a function of one argument, which is quite a limitation when we want to define function accepting multiple arguments. One way to do this would be by extending the λ-calculus to allow the formation of pairs, triples, etc., in which case, say, a three-place function $\lambda x.\, M$ would expect its argument to be a triple. However, it is more convenient to do this by *Currying*.

Let's consider an example. If we want to define a function that accepts two arguments and returns the first, we write $\lambda x.\, \lambda y.\, x$, which literally is a function that accepts an argument and returns a function that accepts another argument and returns the first argument while it drops the second. Let's see what happens when we apply it to two arguments:

$$\begin{aligned}(\lambda x.\, \lambda y.\, x)MN &\xrightarrow{\beta}(\lambda y.\, M)N\\ &\xrightarrow{\beta}M\end{aligned}$$

In general, to write a function with parameters $x_1, \ldots, x_n$ defined by some term N, we can write $\lambda x_1.\, \lambda x_2.\, \ldots \lambda x_n.\, N$. If we apply n arguments to it we get:

$$\begin{aligned}(\lambda x_1.\, \lambda x_2.\, \ldots \lambda x_n.\, N)M_1 \ldots M_n &\xrightarrow{\beta}\\ &\xrightarrow{\beta} ((\lambda x_2.\, \ldots \lambda x_n.\, N)[M_1/x_1])M_2 \ldots M_n\\ &\equiv (\lambda x_2.\, \ldots \lambda x_n.\, N[M_1/x_1])M_2 \ldots M_n\\ &\vdots\\ &\xrightarrow{\beta} P[M_1/x_1] \ldots [M_n/x_n]\end{aligned}$$

The last line literally means substituting M_i for x_i in the body of the function definition, which is exactly what we want when applying multiple arguments to a function.

Photo Credits

Bibliography