Functional Currying Notation

by Sven Nilsen, 2018

In functional programming, it is common to write the following:

$f(a) : B \to C$

$f : A \times B \to C$
$a : A$

This is called "function currying" and can be thought of as auto-constructing a function `f'`

f' := \(a : A) = \(b : B) = f(a, b)

f(a) <=> f'(a)

Path semantics uses functional currying a lot, because of sub-types:

x : [f(a)] c

$x : B \to C$

In addition to left-argument currying, it is common in path semantics to use a right-argument version:

x : [f b] c

$x : A \to C$

When a right-argument version returns `bool`, one can use parentheses like this:

x : [g b] true <=> x : (g b)

$g : A \times B \to \text{bool}$

For example:

x : (> 2)
x : (= 10)

A more complex example:

(x, y) : (f g)

$f : A \times A \to (A \to \text{bool}) \to \text{bool}$
$g : A \to \text{bool}$