Cartesian Products vs Arrows

by Sven Nilsen, 2019

A Cartesian product, or a tuple, is of the following form:

$$(A, B)$$

To represent this geometrically, one would have to use an arrow, because the product is ordered:

The start of the arrow is the first element of the Cartesian product.
The end of the arrow is the second element of the Cartesian product.
The arrow itself, together with the knowledge of its end points, is the same as the Cartesian product.

With other words, an arrow can also be represented as a Cartesian product!

An arrow might be formally defined as the following:

$$\exists\ a, b\ \{\ (a, b)\ \}$$

The truth value of this expression is a Cartesian product, not a boolean nor a number in the unit interval. This might seem strange to people used to classical logic, but nothing stops us from creating a such language in path semantics as long we also create the rules for reasoning in this language.

The expression above talks about a Cartesian product without identifying its elements.
With other words, thinking about an arrow without specifying start and end points.
Hence, it talks about the idea of an arrow itself, no more, no less.

This means that proofs about Cartesian products can be translated into arrows and vice versa.

Information stored in such arrows can be though of as a function:

$$f : A \times B \to T$$

When you draw more than one arrow between two points, one can think about this as adding more information to a single arrow. A single function suffices no matter how many arrows there are. Arrows between arrows require the type system to create a tuple (Cartesian product) from two existing types.

Every definition of a data structure can be translated into a corresponding geometric figure.
For example, a list is a dynamic extension of associative Cartesian products. The geometric figure that corresponds to a list is a path, created by composing arrows.