Intentional Paths

by Sven Nilsen, 2017

An intentional path is a property of a function that can vary for logically equivalent cases. Intentional paths are used to reason about performance and contextual embodiment of functions.

Assume some function \`f\`:

$$f : A \rightarrow B$$

A partial function pair has the type:

$$(x, f(x)) : (A, f(A))$$

By mapping partial function pairs to some cost type, one can reason about the embodiment of a function, since all such maps of the same type are about logically equivalent functions:

$$g : (A, f(A)) \rightarrow C$$

This means that the cost must be determined by the function input. The \`g\` function is called an “embodiment” of \`f\`. When optimizing the performance of a function, one logically operates on the embodiments by combining them and constructing new ones. This is used to create functions that make predictions about the cost of manipulating source code.

An input intentional path maps the input of a function to some cost type. This is a way to simplify the embodiment to a form that is easier to reason about:

$$f^g : A \rightarrow C$$

$$|f^g| : C$$

Because input intentional paths are just functions, one can use standard path semantical notation, such as domain constraints. The “length” of a constrained input intentional path is often used to compare the overall performance of two different algorithms:

$$|f^{g0}\{[even]\ true\}| < |f^{g1}\{[even]\ true\}]|$$

The output intentional path measures the average cost for inputs that return the same value. It is often used in problem solving:

$$g^f : f(A) \rightarrow C$$

$$g^f := \backslash(x : f(A)) = |f^g\{[f]\ x\}| \ / \ |[f]\ x|$$

An output intentional path is distinguished from an input intentional path by its type, so specifying \`$g^f$\` without their types is ambiguous.