Path Semantics Cheat Sheet

by Sven Nilsen, 2017

Path semantics is used to study functions and how they are related to each other. The notation makes it easier to wrap one's head around high order concepts that are used to derive algorithms or do theorem proving.

Symmetric Path

$g(f(a, b)) = h(g(a), g(b))$ ⟺ $f[g] <=> h$

Equations are useful but sometimes hard to spot the patterns. A lot of equations secretly tell something about functions.

This is called a "symmetric path". `h` is the path of `f` by `g`, because it predicts how the property `g` behaves in `f`.

Asymmetric Path

$g_n(f(a, b)) = h(g_0(a), g_1(b))$ ⟺ $f[g_0 \times g_1 \to g_n] <=> h$ ⟺ $f[g_{i \to n}] <=> h$

It gets even harder to spot patterns in equations when all the functions are different.

This is called an "asymmetric path". `h` is the path of `f` by $g_0 \times g_1 \to g_n$, because it predicts how this behaves in `f`.

Asymmetric notation uses indices that can be erased to get symmetric paths. This notation makes it easier to work with higher order concepts.

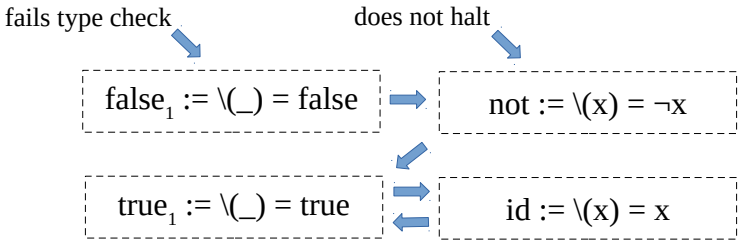
Existential Path

$\exists x \{ g(x) = b \}$ ⟺ $(\exists g)(b) = true$ ⟺ $b : [\exists g]\ true$

It only makes sense to talk about `g` returning `b` if there exists some input `x` that produces `b`.

This is called an "existential path". It returns `true` when its function returns some output and `false` otherwise.

One can think of this as `b` having a sub-type such that the existential path of `g` returns `true`. This makes `b` consistent.



Every double-existential path ∃∃g is one of 4 functions. Strangely, some functions can be thought of as failing type checking or do not terminate on any input. All "normal" functions have double-existential paths `true`$_1$ or `id`. Those who got `id` are structure-preserving.

Every existential path returns a `bool` which is a small type, so there are many functions that share the same existential paths. For example, by rearranging the outputs of a function, its permutations all point to the same existential path. One output can also be repeated more or fewer times but not zero.

Probabilistic Existential Path

$|x : [g]\ b| / |X| > 0.1$ ⟺ $(\exists_p g)(b) > 0.1$ ⟺ $b : [\exists_p g]\ (> 0.1)$

The ratio of how many times a function returns a value to the number of possible inputs. In this case this is more than 10%.

This is called a "probabilistic existential path". It returns how often that function returns some output, assuming finite input and uniform distribution.

One can think of this as `b` having a sub-type such that `g` returns it more often than 10%.

Probabilistic Path

$$f[g_{i \to n}]_p := \backslash(b_{i \to n} : [] \wedge [len]\ |g_{i \to n}|, p : [real] \wedge [len]\ |g_i|) = \sum j\ 2^{|g_i|} \{ (\exists_p(g_n \cdot f)\{2^{[g_i]}\ b_i\}(\beta_j))(b_n) \cdot \prod k\ |g_i| \{ \beta_{jk}((p_k\text{-}(\exists_p g_{ik})(b_k))/(1\text{-}(\exists_p g_{ik})(b_k))) \} \}$$

A probabilistic path finds sub-type probability of output from sub-type probability of input. It is an interesting function. :)