Path Semantics Cheat Sheet

by Sven Nilsen, 2017

Symmetric Path

g(f(a, b)) = h(g(a), g(b))

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

$f[g] \ll h$

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

Asymmetric Path

$$g_n(f(a, b)) = h(g_0(a), g_1(b))$$

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

$f[g_0 \times g_1 \rightarrow g_n] <=> h$

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

$f[g_{i\rightarrow n}] \le h$

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

$$\exists x \{ g(x) = b \}$$

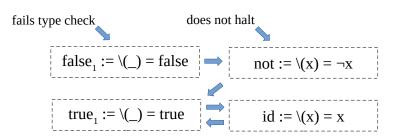
It only makes sense to talk about `g` returning `b` if there exists some input `x` that produces `b`.

$$(\exists g)(b) = true$$

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

b : [∃g] true

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



Every double-existential path $\exists\exists g$ 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_i` or `id`. Those who got `id` are surjective, `true_i` are injective.

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

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

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.

 $b: [\exists_p g] (> 0.1)$

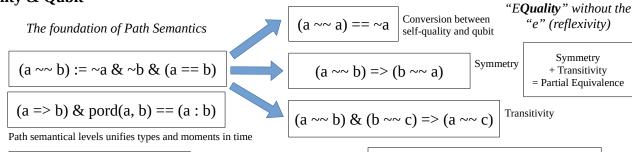
One can think of this as 'b' having a subtype such that 'g' returns it more often than 10%.

Probabilistic Path

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

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

Quality & Qubit



$$(a == b)^{true} => (\sim a == \sim b) \qquad \begin{array}{c} \text{Tautological} \\ \text{congruence of qubit} \end{array} \qquad \begin{array}{c} (a : A) \& (b : B) => \\ (a \sim \sim b) : (A \sim \sim B) \end{array}$$

$$(a == b)^{true} \& (a \sim \sim c) => (b \sim \sim c) \qquad \begin{array}{c} \text{Tautological} \\ \text{congruence of quality} \end{array} \qquad \begin{array}{c} \text{The Core Axiom of Path Semantics} \\ \text{Usually, one uses `(a \sim b) => (A \sim B)` because one can prove the above using path semantical levels} \end{array}$$

congruence of quality

can prove the above using path semantical levels

Lifting equality to quality

$$(a == b) \& sd(a, b) => (a \sim b)$$

$$sd(a, b) => sd(b, a)$$
Symmetry

Equality can be lifted to quality if one can prove that the two sides are symbolic distinct. Symbolic distinctness is a form of propositional infinity. There are infinite axioms, e.g. `sd(a', b')` when `a` and 'b' are symbols.