Entangled Functions

by Sven Nilsen, 2018

An entangled function is one that depends on more than one of the same variable for multiple inputs. All entangled functions can be rewritten as a function of fewer variables. Disentangled functions can not be rewritten as a function of fewer variables.

For example:

(\(a) = mul(a, a)) <=> (\(a) = pow(a, 2)) <=> (pow 2)

In asymmetric path semantical notation using cross argument domain constraint, then picking the first argument and erasing the second with the `unit` function:

mul{(=)}[fst × unit → id] <=> (pow 2)

It is easier to use normal equation form (notice that is an entangled function itself):

mul(a, a) = pow(a, 2)

What makes entangled function interesting is that they change function identity. Like all domain constraint, when input is filtered, the path semantical statements that are true about the function changes, and therefore the identity of the function changes too.

mul(a, -a) = -pow(a, 2)
mul(a, 1/a) = if a == 0 {±∞} else {1}   (projective reals)

In deterministic path semantics, it is not possible to create a gradual transition from a disentangled function to an entangled function. The arguments either share information, or they do not. Here is an entangled function that share a tribit:

mul(a, sign(a) · abs(b)) = abs(mul(a, b))

It is not the case that no disentangled function can be written as an expression of shared variables. For example, the following seemingly entangled function at first sight is actually a disentangled function:

and(a, or(not(a), b)) = and(a, b)

Sometimes more than one variable is erased by the constraint:

and(a, and(not(a), b)) = false