Sub-Type Aliasing

by Sven Nilsen, 2017

In this paper I show that path semantics with constrained functions has a natural proof that corresponds to sub-type aliasing. Complex type signatures can be shortened down significantly without becoming ambiguous. This technique does not require explicit definitions of sub-type aliases, but instead one can directly use the definition of the sub-type.

This technique is best illustrated with a real world example: A cyclic group can be represented as a matrix containing only 1s and 0s where there each column and each row contains only one `1`:

m : matrix ∧ [dim] [eq] true ∧ [cyclic_group] true
cyclic_group : matrix ∧ [dim] [eq] true → bool

By associating the sub-type of `cyclic_group` as the default and largest sub-type, one can write:

m : [cyclic_group] true

A shorthand version, which is compatible with the syntax for defining a new type:

m : cyclic_group

The rest of the paper is proving the soundness of this technique. From reduction of proofs with multiple constraints:

a : [f] b ∧ [g] c <=> a : [f{[g] c}] b ∧ [g{[f] b}] c

f : A → B
g : A → C

To check for consistency it is sufficient to check either case, since one implies the other:

b : [∃f{[g] c}] true <=> c : [∃g{[f] b}] true

∃f{[g] c} : B → bool
∃g{[f] b} : C → bool

Something interesting happens when adding a new assumption:

f{[g] c} <=> ∀f

b : [∃f{[g] c}] true <=> b : [∃f{∀f}] true <=> b : [∃f] true
a : [f] b ∧ [g] c <=> a : [f] b

Therefore, `f` has taken on the role of defining the whole sub-type, such that `[g] c` can be eliminated.