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`:

$$m : \text{matrix} \wedge [\text{dim}]\ [\text{eq}]\ \text{true} \wedge [\text{cyclic\_group}]\ \text{true}$$
$$\text{cyclic\_group} : \text{matrix} \wedge [\text{dim}]\ [\text{eq}]\ \text{true} \rightarrow \text{bool}$$

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

$$m : [\text{cyclic\_group}]\ \text{true}$$

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

$$m : \text{cyclic\_group}$$

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

$$a : [f]\ b \wedge [g]\ c \iff a : [f\{[g]\ c\}]\ b \wedge [g\{[f]\ b\}]\ c$$

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

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

$$b : [\exists f\{[g]\ c\}]\ \text{true} \quad \iff \quad c : [\exists g\{[f]\ b\}]\ \text{true}$$

$$\exists f\{[g]\ c\} : B \rightarrow \text{bool}$$
$$\exists g\{[f]\ b\} : C \rightarrow \text{bool}$$

Something interesting happens when adding a new assumption:

$$[g]\ c \iff \forall f$$

$$b : [\exists f\{[g]\ c\}]\ \text{true} \iff b : [\exists f\{\forall f\}]\ \text{true} \iff b : [\exists f]\ \text{true}$$
$$a : [f]\ b \wedge [g]\ c \iff a : [f]\ b$$

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