Covariant generics. Patterns of danger. How to detect them statically?

It applies only to the entities of the ref type!

// Synthetic example of the issue

unit A[G] // We have generic unit A

attr: G := /\* It has an attribute attr and we can directly assign the value of this attribute using default setter \*/

end

a: A[T] // DynamicTypeSet: {…, A[U], …}

b: A[U] // U conforms to T (U is a descendant of T, U is derived from T)

a := b /\* every assignment leads to potential extension of the dynamic types set for the assignment target. The target is to be named entity, as well as the source too!!! \*/

a.attr := new T

attr is b.attr /\* What is the type of attr? Static type of attr is equal to the static type of b.attr -> U but dynamic is T. So, the breakage of type system occurred \*/

a := new A[U] // such assignment is always type-safe !!! As we have no direct way how to access A[U]

if a is A[U] do /\* this type check must dynamically return false if polymorphic assignment was not executed \*/

attr is a.attr // attr is of static type U

end // if

// More explicit example

a: Array [Any] // a is initialized as an empty array

b: Array [String] is (“s1”, “s2”, “s3”)

a := b // That is the place which creates the basis for a headache

a(1) := Integer.5 // Put integer value into the 1st element

a(2) := Real.3.1415492653589 // Put real value into the 2nd element

a(3) := (“string1”, Boolean.true) // Put tuple into the 2rd one

var myString: String is b(1)

myString := b(2)

myString := b(3)

Call is valid if it is static-type valid and valid for every dynamic type of the call target

target.foo (argument1, argument2, … , argumentn)

static type of target is D0

Dynamic type set of the target is D1, … Dm D0 can be part of this set or not. (D0 can be abstract type for example)

Static-type validity