Proposed Resolution for Valarray Constructors

Gabriel Dos Reis

Abstract
This short note proposes a simple resolution to Library Issue 630.

1 The Issue

Library Issue 630 is about the apparent inconsistency between two requirements on value arrays:

1. a valarray can be instantiated only with numeric types, e.g. satisfying the properties listed in §26.1/1;

2. assigning a valarray object to another valarray object of unequal length results in undefined behaviour. Consequently, a valarray instantiation does not meet the requirements of §26.1/1 or numeric types.

To resolve this tension, it has been suggested to allow assignment between valarray objects of possibly unequal length. This note suggests a different resolution.

2 Proposed Resolution

There are several observations to be made. First, even if assignment between valarray objects of unequal lengths is allowed, valarray instantiations would
still not satisfy the numeric type requirements, for copy construction may throw an exception whereas numeric types are not allowed to throw exceptions.

Second, the idiomatic way to construct array of value arrays it to construct one “big” (one-dimensional) valarray object, then make a gslice_array view out of if using gslice.

Third, there is a fundamental issue of consistency of the interface of the valarray component. What to be said for all operations involving more than one valarray objects? This is a slippery slope. It is not clear that the original issue really is fundamental, and warrants surgery to the general, uniform, descriptions of standard operations on valarrays.

Therefore we suggest that footnote 269 does not mention valarray instantiations. Furthermore, if it really is desired to support valarray instantiated with instantiations of valarray — a non-idiomatic use of the library — then it would suffice alter §26.5.2.2/1 as follows

1 If the length of the *this is zero, then the *this array is resized to match the length of the array argument. Otherwise, the resulting behaviour is undefined if the length of the argument array is not equal to the length of *this array. Each element of the *this array is assigned the value of the corresponding element of the argument array. The resulting behaviour is undefined if the length of the argument array is not equal to the length of *this array.