Document number P0716R0

Date 2017-06-19

Authors Richard Smith <richard@metafoo.co.uk>
Andrew Sutton <asutton@uakron.edu>

Audience Evolution

Unified concept definition syntax

Introduction

At Kona, P0324R0 was presented and Evolution gave the following guidance (given in the usual strongly in favor | in favor | neutral | against | strongly against form):

Explore removing the distinction between function-like and variable-like concept definitions?
24 | 23 | 1 | 0 | 0
Explore removing the bool?
20ish | 10ish | 5 | 0 | 0

However, no vote was taken to forward P0324R0 to core. Discussion since Kona has indicated that there is still a very high level of support for this change. This paper requests that such a vote be taken, and suggests a specific approach from those proposed by P0324R0.

Rationale

We refer the reader to P0324R0 for detailed rationale for the proposed change, but would highlight the following points:

- With both function and variable templates permitted, the user of a concept must know which form is used: implementation details leak into the interface
- Function and variable declarations carry a large amount of baggage (linkage, declarator syntax, forward declarations, type specifiers, various initialization syntaxes, initialization order issues, destruction semantics, and so on) that do not make sense for concepts. We should not burden concepts with this baggage.

Background

The original concepts proposal developed in 2011 defined functions in terms of functions, using `concept` as a declaration specifier. Variable templates were added to C++14 in the 2013

Bristol meeting. Several committee members noted that the the parentheses on concepts could be omitted if the declaration specifier also applied to variable templates. That change was accepted, ultimately leading to issues discussed in P0324R0.

Approach

We propose restricting to a single concept definition syntax, similar to the current syntax but with the “bool” removed and with the other complexities of variable declaration syntax similarly excised. Specifically, the only permitted syntax would be:

```
template < template-parameter-list >
concept identifier = constraint-expression ;
```

For simplicity of exposition, we propose following a path similar to P0324R0’s Approach 3 or 5: define a separate grammar production for concepts instead of reusing the function / variable declaration grammar. However, such an approach is intended to be formally equivalent to defining a concept as being a variable template that is implicitly declared to be ‘constexpr bool’ and where grammatical complexities beyond the syntax above are disallowed (some, but not all, of these restrictions already exist in the Concepts TS).

We propose removing the ability to overload concepts on differing *template-parameter-list*s. This removal is not fundamental: with the above reformulation of concepts as being distinct from variables, we could permit overloading without needing to introduce overloading on template parameters to variable templates, but there seems to be little support for retaining the ability to overload concepts.

We propose one additional change, mentioned in footnote 4 of P0324R0: we propose that an *id-expression* naming a specialization of a concept (such as `Trivial<int>`) be a prvalue, rather than a lvalue. This means that concept specializations behave like manifest constants, not like variables, and matches the behavior of other manifest constants, such as literals, (non-reference) non-type template arguments, and enumerators. This also avoids the need for such concept specializations to be emitted as data in executables.

Interaction with Ranges TS

The Ranges TS currently does make use of concept overloading for several concepts (such as providing both `EqualityComparable<T>` and `EqualityComparable<T, U>` for determining whether a type is equality comparable to itself and to another type). It also exclusively uses function concepts, in order to serve the dual goals of permitting overloading and avoiding concept users from needing to know whether a particular concept is a function concept or a variable concept.

A separate paper from Eric Niebler will propose switching the Ranges TS to non-overloaded variable concepts, and the primary authors of the Ranges TS have raised no objection to unifying the concept definition syntax as described here.