X3J16/96-0154
WG21/N0972
J. Stephen Adamczyk
jsa@edg.com

# WP Changes for Overload Resolution (reference binding and similar small issues)

In 8.5.3 [dcl.init.ref], paragraph 7, after the first sentence, add the sentence "In this case the reference is said to *bind directly* to the initializer expression."

In 13.3.3.1 [over.best.ics], replace paragraph 5 by

> For the case where the parameter type is a reference, see _over.ics.ref_.

Following 13.3.3.1 [over.best.ics], paragraph 5, insert the following new paragraph:

> When the parameter type is not a reference, the implicit conversion sequence models a copy-initialization of the parameter from the argument expression. The implicit conversion sequence is the one required to convert the argument expression to an rvalue of the type of the parameter. [Note: when the parameter has a class type, this is a conceptual conversion defined for the purposes of this clause; the actual initialization is defined in terms of constructors and is not a conversion.] Any difference in top-level cv-qualification is subsumed by the initialization itself and does not constitute a conversion. [Example: a parameter of type `A` can be initialized from an argument of type `const A`. The implicit conversion sequence for that case is the identity sequence; it contains no "conversion" from `const A` to `A`.] When the parameter has a class type and the argument expression is an rvalue of the same type, the implicit conversion sequence is an identity conversion. When the parameter has a class type and the argument expression is an lvalue of the same type, the implicit conversion sequence is an lvalue-to-rvalue conversion. When the parameter has a class type and the argument expression type is a rvalue of a derived class type, the implicit conversion sequence is a Conversion from the derived class to the base class. [Note: there is no such standard conversion; this Conversion exists only in the description of implicit conversion sequences.] When the parameter has a class type and the argument expression is an lvalue of a derived class type, the implicit conversion sequence is an lvalue-to-rvalue conversion followed by a derived-to-base conversion. A derived-to-base Conversion has Conversion rank (_over.ics.scs_).

In 13.3.3.1.4 [over.ics.ref], replace paragraph 1 by

> When a parameter of reference type binds directly (_dcl.init.ref_) to an argument expression, the implicit conversion sequence is the identity conversion, unless the argument expression has a type that is a derived class of the parameter type, in which case the implicit conversion sequence is a derived-to-base Conversion. If the parameter binds directly to the result of applying a conversion function to the argument expression, the implicit conversion sequence is a user-defined conversion sequence (_over.ics.user_), with the second standard conversion sequence either an identity conversion or, if the conversion function returns an entity of a type that is a derived class of the parameter type, a derived-to-base Conversion.
>
> [move to here the example following the first bullet of paragraph 4 of this section.]
>
> When a parameter of reference type is not bound directly to an argument expression, the conversion sequence is the one required to convert the argument expression to the underlying type of the reference according to _over.best.ics_. Conceptually, this conversion sequence corresponds to copy-initializing a temporary of the underlying type with the argument

> expression. Any difference in top-level cv-qualification is subsumed by the initialization itself and does not constitute a conversion.

Remove all of 13.3.3.1.4 [over.ics.ref], paragraph 4, except the text of the last bullet (which becomes the complete paragraph). As noted, the example here is moved to earlier in the section.

In 13.3.3.2 [over.ics.rank], paragraph 3, first bullet, first sub-bullet, after "`S1` is a proper subsequence of `S2`" add "(comparing the conversion sequences in the canonical form defined by _over.ics.scs_; the identity conversion is considered to be a subsequence of any non-identity conversion sequence)".

In 13.3.3.2 [over.ics.rank], paragraph 3, first bullet, fourth sub-bullet (beginning "`S1` and `S2` are reference bindings"), replace the first sentence by

> `S1` and `S2` are reference bindings (_dcl.init.ref_), and the underlying types of the references are the same type except for top-level cv-qualifiers, and the underlying type of the reference initialized by `S2` is more cv-qualified than the underlying type of the reference initialized by `S1.`