Core II WP Changes

Core Issue 668 (const string literals and how they participate in overload resolution):

Add to the end of 4.2 [conv.array] paragraph 2:

For the purpose of ranking in overload resolution (_over.ics.scs_), this conversion is considered an array-to-pointer conversion followed by a qualification conversion. [Example: "abc" is converted to "pointer to const char" as an array-to-pointer conversion, and then to "pointer to char" as a qualification conversion.]

Core Issue 670 (multi-level cv-qualification on pointer operands):

Add to 4.4 [conv.qual] paragraph 3, after "const, volatile, const volatile, or nothing", starting a new paragraph:

The n-tuple of cv-qualifications after the first in a pointer type, e.g., \(cv_{1,1}, cv_{1,2}, \ldots, cv_{1,n}\) in the pointer type \(T_1\), is called the cv-qualification signature of the pointer type.

Replace 4.4 [conv.qual] paragraph 5 by:

Two multi-level pointer to member types or two multi-level mixed pointer and pointer to member types \(T_1\) and \(T_2\) are similar if there exists a type \(T\) and integer \(N > 0\) such that:

\[
T_1 \text{ is } cv_{1,0} P_0 \text{ to } cv_{1,1} P_1 \text{ to } \ldots \text{ to } cv_{1,n-1} P_{n-1} \text{ to } cv_{1,n} T
\]

and

\[
T_2 \text{ is } cv_{2,0} P_0 \text{ to } cv_{2,1} P_1 \text{ to } \ldots \text{ to } cv_{2,n-1} P_{n-1} \text{ to } cv_{2,n} T
\]

For similar multi-level pointer to member types and similar multi-level mixed pointer and pointer to member types, the rules for adding cv-qualifiers are the same as those used for similar pointer types.

In 5.9 [expr.rel] paragraph 2, change
Pointer conversions ... are performed on pointer operands ... to bring them to the same type, which shall be a cv-qualified or cv-unqualified version of the type of one of the operands.

to

Pointer conversions (_conv.ptr_) and qualification conversions (_conv.qual_) are performed on pointer operands (or on a pointer operand and a null pointer constant) to bring them to their composite pointer type. If one operand is a null pointer constant, the composite pointer type is the type of the other operand. If one of the operands has type "pointer to cv1 void", the other has type "pointer to cv2 T", and the composite pointer type is "pointer to cv3 void", where cv3 is the union of cv1 and cv2. Otherwise, the composite pointer type is a pointer type similar (_conv.qual_) to the type of one of the operands, with a cv-qualification signature (_conv.qual_) that is the union of the cv-qualification signatures of the operand types.

Change 5.10 [expr.eq] paragraph 2 from

... are performed to bring them to the same type, ... the type of one of the operands.

to

... are performed to bring them to a common type. If one operand is a null pointer constant, the common type is the type of the other operand. Otherwise, the common type is a pointer to member type similar (_conv.qual_) to the type of one of the operands, with a cv-qualification signature (_conv.qual_) that is the union of the cv-qualification signatures of the operand types.

Change 5.16 [expr.cond] paragraph 5 bullet 3 from

... to bring them to a common type ... either the second or the third operand.

to

... to bring them to their composite pointer type (_expr.rel_).

Core Issue 684 (comparison of conversion sequences on multi-level pointers):

Change 13.3.3.2 [over.ics.rank] paragraph 3 bullet 1 sub-bullet 3 to:

S1 and S2 differ only in their qualification conversion and yield similar types T1 and T2 (_conv.qual_), respectively, and the cv-qualification signature of type T1 is a proper subset of the cv-qualification signature of type T2,

Core Issue 685 (ambiguity in old-style cast):

Add to 5.4 [expr.cast] at the end of paragraph 5, before the example:
If a conversion can be interpreted in more than one way as a `static_cast` followed by a `const_cast`, the conversion is ill-formed.

Core Issue 645b (lvalue-to-rvalue conversion on void expressions):

Add to the end of 5.2.9 \[expr.static.cast\] paragraph 4:

The lvalue-to-rvalue (_conv.lval_), array-to-pointer (_conv.array_) and function-to-pointer (_conv.func_) standard conversions are not applied to the expression.

Add to 5.18 \[expr.comma\] paragraph 1 after the first sentence:

The lvalue-to-rvalue (_conv.lval_), array-to-pointer (_conv.array_) and function-to-pointer (_conv.func_) standard conversions are not applied to the left expression.

Add to 6.2 \[stmt.expr\] paragraph 1, following the syntax:

The expression is evaluated and its value is discarded. The lvalue-to-rvalue (_conv.lval_), array-to-pointer (_conv.array_) and function-to-pointer (_conv.func_) standard conversions are not applied to the expression.

Core Issue 662 (cv-qualifiers on call to object of class type):

Add to 13.3.1.1.2 \[over.call.object\], paragraph 2, after the syntax:

where `cv-qualifier` is the same cv-qualification as, or a greater cv-qualification than, `cv`, and where ...