Resolution for core issues 1207 and 1017

Proposed resolution:

- Change 5.1.1 expr.prim.general p2 as indicated and insert the new paragraphs 3, 4, and 5 after that (and move the example from p2 to the end of the new p5):

  2. The keyword this names a pointer to the object for which a non-static member function (9.3.2 class.this) is invoked or a non-static data member's initializer (9.2 class.mem) is evaluated. The keyword this shall be used only inside the body of a non-static member function (9.3 class.mfct) of the nearest enclosing class or in a brace-or-equal-initializer for a non-static data member (9.2 class.mem). The type of the expression is a pointer to the class of the function or non-static data member, possibly with cv-qualifiers on the class type. The expression is a prvalue.

  3. If a declaration declares a member function or member function template of a class X, the expression this is a prvalue of type "pointer to cv-qualifier-seq X" between the optional cv-qualifier-seq and the end of the function-definition, member-declarator, or declarator. It shall not appear before the optional cv-qualifier-seq and it shall not appear within the declaration of a static member function (although its type and value category are defined within a static member function as they are within a non-static member function). [ Note: this is because declaration matching does not occur until the complete declarator is known. — end note ] Unlike the object expression in other contexts, *this is not required to be of complete type for purposes of class member access (5.2.5 expr.ref) outside of the member function body. [ Note: Only class members declared prior to the declaration are visible. —end note ] [Example:

```cpp
struct A {
    char g();
     template<class T> auto f(T t)->decltype(t + g())
    {
        return t + g();
    }
};

template auto A::f(int t)->decltype(t + g());
```

—end example]

4. Otherwise, if a member-declarator declares a non-static data member (9.2 class.mem) of a class X, the expression this is a prvalue of type "pointer to X" within the optional brace-or-equal-initializer. It shall not appear elsewhere in the member-declarator.

5. The expression this shall not appear in any other context. [Example:

```cpp
class Outer {
    int a[sizeof(*this)]; // error: not inside a member function
    unsigned int sz = sizeof(*this); // OK: in brace-or-equal-initializer

    void f() {
        int b[sizeof(*this)]; // OK

        struct Inner {
            int c[sizeof(*this)]; // error: not inside a member function of Inner
        }
    }
};
```

—end example]

- Change 5.1.1 expr.prim.general, old-paragraph-10, as indicated.

  10. An id-expression that denotes a non-static data member or non-static member function of a class can only be used:

   - as part of a class member access (5.2.5 expr.ref) in which the object-expression refers to the member's class [Footnote: This also applies when the object expression is an implicit (*this) (9.3.1 class.mfct.non-static). — end footnote ] or
   - a class derived from that class, or
   - to form a pointer to member (5.3.1 expr.unary.op), or
3. When an id-expression (5.1 expr.prim) that is not part of a class member access syntax (5.2.5 expr.ref) and not used to form a pointer to member (5.3.1 expr.unary.op) is used in the body of a non-static member function of class X a member of class X in a context where this can be used (5.1.1 expr.prim.general), if name lookup (3.4 basic.lookup) resolves the name in the id-expression to a non-static non-type member of some class C, and if either the id-expression is potentially evaluated or C is X or a base class of X, the id-expression is transformed into a class member access expression (5.2.5 expr.ref) using (*this) (9.3.2 class.this) as the postfix-expression to the left of the . operator. [Note: if C is not X or a base class of X, the class member access expression is ill-formed. — end note] Similarly during name lookup, when an unqualified-id (5.1) used in the definition of a member function for class X resolves to a static member, an enumerator or a nested type of class X or of a base class of X, the unqualified-id is transformed into a qualified-id (5.1) in which the nested-name-specifier names the class of the member function. [Example: [...] — end example]