Document Numbers: X3J16/97-0063R1
WG21/N1101R1
Date: July 17, 1997
Reply To: Bill Gibbons
bill@gibbons.org

Core-III Changes from CD2 Ballot Comments, Part 2

Introduction

The following is a comparison of original and modified troff files for working paper changes for Core III, for clauses 3, 14 and 15.

Clause 3 [basic]

```
File #1: :original:basic
File #2: :modified:basic

Extra lines in 2nd before 1349 in 1st (File ":original:basic"; Line Δ1349; File
":modified:basic"; Line 1349:1350)
1347     .H3 "Argument-dependent name lookup" basic.lookup.koenig
1348     .ix "argument-dependent lookup"
--------------------------------------------------------------------------------
1349     .\" L7003 USA Core3 1.12 public comment 23 / 3.4.2 [basic.lookup.koenig]
1350     .\" and 14.6.5 [temp.inject]   Friend name lookup, as previously agreed.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1349     .P
1350     When an unqualified name is used as the


Nonmatching lines (File ":original:basic"; Line 1354:1355; File ":modified:basic";
Line 1356:1359)
1352     in a function call (_expr.call_),
1353     other namespaces not considered during the usual unqualified look up
--------------------------------------------------------------------------------
1354     (_basic.lookup.unqual_) may be searched; this search depends on the types of
1355     the arguments.

1356     (_basic.lookup.unqual_) may be searched, and namespace-scope friend function
1357     declarations (_class.friend_) not otherwise visible may be found.
1358     These modifications to the search depend on the types of the arguments
1359     (and for template template arguments, the namespace of the template
argument).
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1356     .P
1357     For each argument type


Nonmatching lines (File ":original:basic"; Line 1360:1364; File ":modified:basic";
Line 1364:1368)
1358     .CW T
1359     in the function call, there is a set of zero or more associated namespaces
--------------------------------------------------------------------------------
1360     to be considered.
1361     The set of namespaces is determined entirely by the types of the function
1362     arguments.
1363     Typedef names used to specify the types do not contribute to this set.
1364     The set of namespaces are determined in the following way:

1364     and a set of zero or more associated classes to be considered.
1365     The sets of namespaces and classes is determined entirely by the types of the
1366     function arguments (and the namespace of any template template argument).
1367     Typedef names used to specify the types do not contribute to this set.
1368     The sets of namespaces and classes are determined in the following way:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1365     .LI
1366     If


Nonmatching lines (File ":original:basic"; Line 1368; File ":modified:basic"; Line
1372:1373)
1366     If
1367     .CW T
--------------------------------------------------------------------------------
1368     is a fundamental type, its associated set of namespaces is empty.

1372     is a fundamental type, its associated sets of namespaces and
1373     classes are both empty.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1369     .LI
1370     If


Nonmatching lines (File ":original:basic"; Line 1372:1374; File ":modified:basic";
Line 1377:1380)
1370     If
1371     .CW T
--------------------------------------------------------------------------------
1372     is a class type, its associated namespaces are the namespaces
1373     in which the class and its direct and indirect base classes are
1374     defined.

1377     is a class type, its associated classes are the class itself
1378     and its direct and indirect base classes.
1379     Its associated namespaces are the namespaces in
1380     which its associated classes are defined.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1375     .LI
1376     If


Nonmatching lines (File ":original:basic"; Line 1379; File ":modified:basic"; Line
1385:1387)
1377     .CW T
1378     is a union or enumeration type, its associated namespace is the
--------------------------------------------------------------------------------
1379     namespace in which it is defined.

1385     namespace in which it is defined.  If it is a class member, its
1386     associated class is the member's class; else it has no associated
1387     class.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
1380     .LI
1381     If


Nonmatching lines (File ":original:basic"; Line 1389; File ":modified:basic"; Line
1397)
1387     or an array of
1388     .CW U ,
--------------------------------------------------------------------------------
1389     its associated namespaces are the namespaces associated with

1397     its associated namespaces and classs are those associated with
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1390     .CW U .
1391     .LI


Nonmatching lines (File ":original:basic"; Line 1394:1395; File ":modified:basic";
Line 1402:1403)
1392     If
1393     .CW T
--------------------------------------------------------------------------------
1394     is a pointer to function type, its associated namespaces are
1395     the namespaces associated with the function parameter types and the
namespaces

1402     is a pointer to function type, its associated namespaces and classes are
1403     those associated with the function parameter types and those
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1396     associated with the return type.
1397     .LI


Nonmatching lines (File ":original:basic"; Line 1402:1403; File ":modified:basic";
Line 1410:1411)
1400     is a pointer to a member function of a class
1401     .CW X ,
--------------------------------------------------------------------------------
1402     its associated namespaces are the namespaces associated with the function
1403     parameter types and return type, together with the namespaces associated with

1410     its associated namespaces and classes are those associated with the function
1411     parameter types and return type, together with those associated with
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1404     .CW X .
1405     .LI


Nonmatching lines (File ":original:basic"; Line 1410:1411; File ":modified:basic";
Line 1418:1419)
1408     is a pointer to a data member of class
1409     .CW X ,
--------------------------------------------------------------------------------
1410     its associated namespaces are the namespaces associated with the member type
1411     together with the namespaces associated with

1418     its associated namespaces and classes are those associated with the member
type
1419     together with those associated with
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1412     .CW X .
1413     .LI


Nonmatching lines (File ":original:basic"; Line 1418:1422; File ":modified:basic";
Line 1426:1432)
1416     is a
1417     .I template-id ,
--------------------------------------------------------------------------------
1418     its associated namespaces are the namespaces in which the template is
defined,
1419     the namespaces
1420     associated with the types of the template arguments provided for template
1421     type parameters (excluding template template parameters), and the namespaces
1422     of any template template arguments.

1426     its associated namespaces and classes are the namespace in which the template
1427     is defined; for member templates, the member template's class;
1428     the namespaces and classes
1429     associated with the types of the template arguments provided for template
1430     type parameters (excluding template template parameters); the namespaces
1431     in which any template template arguments are defined; and the classes in
1432     which any member templates used as template template arguments are defined.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1423     .N[
1424     non-type template arguments do not contribute to the set of associated


Nonmatching lines (File ":original:basic"; Line 1436; File ":modified:basic"; Line
1446:1448)
1434     When considering an associated namespace, the lookup is the same as the
lookup
1435     performed when the associated namespace is used as a qualifier
--------------------------------------------------------------------------------
1436     (_namespace.qual_) except that

1446     (_namespace.qual_) except that:
1447     .LI
1448     Any
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1437     .I using-directive s
1438     in the associated namespace are ignored.


Extra lines in 2nd before 1439 in 1st (File ":original:basic"; Line Δ1439; File
":modified:basic"; Line 1451:1454)
1437     .I using-directive s
1438     in the associated namespace are ignored.
--------------------------------------------------------------------------------
1451     .LI
1452     Any namespace-scope friend functions declared in
1453     associated classes are visible within their respective namespaces even if
1454     they are not visible during an ordinary lookup (_class.friend_).
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1439     .LE
1440     .H3 "Qualified name look up" basic.lookup.qual
```

Clause 14 [template]

```
File #1: :original:template
File #2: :modified:template

Nonmatching lines (File ":original:template"; Line 88:99; File ":modified:template";
Line 88:104)
  86     in that scope.
  87     .P
--------------------------------------------------------------------------------
  88     A non-inline function template, a non-inline member function template,
  89     a non-inline member function of a class template or
  90     a static data member of a class template is
  91     called an
  92     .I exported
  93     template if its definition is preceded by the keyword
  94     .CW export
  95     or if it has been previously declared using the keyword
  96     .CW export
  97     in the same translation unit.  Declaring a class
  98     template exported is equivalent to declaring all of its non-inline function
  99     members, static data members, member classes, and non-inline member templates

  88     .\" L3204 Germany 2 / 14p6
  89     .\" Clarify where the keyword "export" is required and/or allowed.
  90     A namespace-scope declaration or definition of a non-inline function
template,
  91     a non-inline member function template,  a non-inline member function of a
class
  92     template or a static data member of a class template may be preceded by the
  93     .CW export
  94     keyword.  If such a template is defined in the same translation unit in which
  95     it is declared as exported, the definition is considered to be
  96     .I exported.
  97     The first declaration of the template containing the
  98     .CW export
  99     keyword must not follow the definition.
 100     .P
 101     Declaring a class
 102     template exported is equivalent to declaring all of its non-inline function
 103     members, static data members, member classes, member class templates and
 104     non-inline function member templates
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 100     which are defined in that translation unit exported.
 101     .P


Extra lines in 2nd before 319 in 1st (File ":original:template"; Line Δ319; File
":modified:template"; Line 324:325)
 317     .E[
 318     .Cb
--------------------------------------------------------------------------------
 324     .\" L3282 Germany 6 / 14.1/7
 325     .\" Example was wrong; nontype pointer args may not be variables
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 319             template<int a[5]> struct S { /* ... */ };
 320             int v[5];


Nonmatching lines (File ":original:template"; Line 321:323; File
":modified:template"; Line 328)
 319             template<int a[5]> struct S { /* ... */ };
 320             int v[5];
--------------------------------------------------------------------------------
 321             int* p = v;
 322             S<v> x; // fine
 323             S<p> y; // also fine

 328             S<v> x; // fine
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 324     .Ce
 325     .E]


Extra lines in 2nd before 388 in 1st (File ":original:template"; Line Δ388; File
":modified:template"; Line 393:396)
 386     .H2 "Names of template specializations" temp.names
 387     .P
--------------------------------------------------------------------------------
 393     .\" L3332 Germany 8 / 14.2p1 [temp.names] and 14.3p11 [temp.arg]
 394     .\" Change grammar in 14.2 to match text in 14.3p11 which says that
 395     .\" a template-argument-list may be empty.
 396     .\" Fixed in Lajoie's pre-London suggested changes to clause 14.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 388     A template specialization (_temp.spec_) can be referred to by a
 389     .I template-id :


Nonmatching lines (File ":original:template"; Line 449; File ":modified:template";
Line 458:463)
 447     .CW reinterpret_cast
 448     or
--------------------------------------------------------------------------------
 449     .CW const_cast

 458     .\" USA editorial comment, from core3 issue 3.29
 459     .CW const_cast ,
 460     or which encloses the
 461     .I template-argument s
 462     of a subsequent
 463     .I template-id
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 450     is considered nested for the purpose of this description.
 451     .Fe


Extra lines in 2nd before 468 in 1st (File ":original:template"; Line Δ468; File
":modified:template"; Line 482:484)
 466     .E]
 467     .P
--------------------------------------------------------------------------------
 482     .\" L6924 USA Core3 1.2 core-765 / 14.2p4 [temp.names]
 483     .\" Clarify description and example of when the template
 484     .\" keyword is required.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 468     When the name of a member template specialization appears after
 469     .CW .


Nonmatching lines (File ":original:template"; Line 477:479; File
":modified:template"; Line 494:498)
 475     .CW ::
 476     in a
--------------------------------------------------------------------------------
 477     .I qualified-id
 478     that explicitly depends on a template-argument (_temp.dep_),
 479     the member template name must be prefixed by the keyword template.
```

```
 494     .I qualified-id ,
 495     and the postfix-expression or qualified-id
 496     explicitly depends on a template-argument (_temp.dep_),
 497     the member template name must be prefixed by the keyword
 498     .CW template .
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 480     Otherwise the name is assumed to name a non-template.
 481     .E[


Nonmatching lines (File ":original:template"; Line 486:492; File
":modified:template"; Line 505:518)
 484             public:
 485                     template<size_t> X* alloc();
----------------------------------------------------------------------------------
 486             };
 487             void f(X* p)
 488             {
 489                     X* p1 = p->alloc<200>();
 490                             // ill-formed: < means less than
 491
 492                     X* p2 = p->template alloc<200>();

 505                     template<size_t> static X* adjust();
 506             };
 507             template<class T> void f(T* p)
 508             {
 509                     T* p1 = p->alloc<200>();
 510                             // ill-formed: < means less than
 511
 512                     T* p2 = p->template alloc<200>();
 513                             // fine: < starts explicit qualification
 514
 515                     T::adjust<100>();
 516                             // ill-formed: < means less than
 517
 518                     T::template adjust<100>();
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 493                             // fine: < starts explicit qualification
 494             }


Extra lines in 2nd before 541 in 1st (File ":original:template"; Line Δ541; File
":modified:template"; Line 567:577)
 539     .I class-name
 540     (_class_).
----------------------------------------------------------------------------------
 567     .\" L7171 USA Core3 2.8 N1053 issue 8.11 / 14.2 [temp.names]
 568     .\" Specializations are not found by name lookup on their
 569     .\" template-id, since template-id's themselves do not participate
 570     .\" in name lookup.
 571     Even though a
 572     .I template-id
 573     is a name, it is not an identifier and is not found by name lookup.
 574     .N[
 575     However, the template name portion of a template-id is found by
 576     name lookup.
 577     .N]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 541     .H2 "Template arguments" temp.arg
 542     .P


Extra lines in 2nd before 598 in 1st (File ":original:template"; Line Δ598; File
":modified:template"; Line 635:636)
 596     .E]
 597     .P
----------------------------------------------------------------------------------
 635     .\" L3112  Canada 11 / 14.3p3 [temp.arg]
 636     .\" Fixed in Lajoie's pre-London suggested changes to clause 14.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 598     A
 599     .I template-argument


Extra lines in 2nd before 697 in 1st (File ":original:template"; Line Δ697; File
":modified:template"; Line 736:740)
 695     to bring it to the type of its corresponding
 696     .I template-parameter .
----------------------------------------------------------------------------------
 736     .\" L3282 Germany 6 / 14.1p7 [temp.param] and 14.3 [temp.arg]
 737     .\" Implicit array-to-pointer and function-to-pointer conversions on template
args.
 738     .\" L7094 USA Core3 2.2 core-759 / 14.3p6 [temp.arg]
 739     .\" Reference derived-to-base "conversions" on nontype template args;
 740     .\" no longer a problem because the conversions have been removed.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 697     For a non-type
 698     .I template-parameter


Nonmatching lines (File ":original:template"; Line 700:706; File
":modified:template"; Line 744:749)
 698     .I template-parameter
 699     of pointer type,
----------------------------------------------------------------------------------
 700     qualification conversions (_conv.qual_), the derived-to-base pointer
 701     conversions and the pointer conversions to
 702     .CW void*
 703     (_conv.ptr_)
 704     are applied to an expression used as a
 705     .I template-argument
 706     to bring it to the type of its corresponding

 744     qualification conversions (_conv.qual_),
 745     the array-to-pointer conversion (_conv.array_) and the function-to-pointer
 746     conversion (_conv.func_)
 747     are applied to an expression used as a
 748     .I template-argument
 749     as needed to bring it to the type of its corresponding
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 707     .I template-parameter .
 708     For a non-type


Nonmatching lines (File ":original:template"; Line 711:712; File
":modified:template"; Line 754)
 709     .I template-parameter
 710     of pointer to member type,
----------------------------------------------------------------------------------
 711     qualification conversions (_conv.qual_) and the base-to-derived pointer to
 712     member conversions (_conv.mem_)

 754     qualification conversions (_conv.qual_)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 713     are applied to an expression used as a
 714     .I template-argument
```

```
Nonmatching lines (File ":original:template"; Line 720:723; File
":modified:template"; Line 762:765)
 718     .I template-parameter
 719     of reference type,
--------------------------------------------------------------------------------
 720     the expression used as a
 721     .I template-argument
 722     shall be reference-compatible (_dcl.init.ref_) with the type of the
 723     corresponding

 762     qualification conversions (_conv.qual_)
 763     are applied to an expression used as a
 764     .I template-argument
 765     to bring it to the underlying non-reference type of its corresponding
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 724     .I template-parameter .
 725     .E[


Nonmatching lines (File ":original:template"; Line 731:736; File
":modified:template"; Line 773:780)
 729             X<ai> xi;  // array to pointer and qualification conversions
 730
--------------------------------------------------------------------------------
 731             struct Base { /* ... */ };
 732             struct Derived : Base { /* ... */ };
 733             template<Base& b> struct Y { /* ... */ };
 734             Derived d;
 735             Y<d> yd;   // template-argument is reference-compatible
 736                        // with template-parameter

 773             struct Y { /* ... */ };
 774             template<const Y& b> struct Z { /* ... */ };
 775             Y y;
 776             Z<y> z;   // qualification conversion
 777
 778             template<int (&pa)[5]> struct W { /* ... */ };
 779             int b[5];
 780             W<b> w;   // no conversion
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 737     .Ce
 738     .E]


Extra lines in 2nd before 774 in 1st (File ":original:template"; Line Δ774; File
":modified:template"; Line 818:821)
 772     .E]
 773     .P
--------------------------------------------------------------------------------
 818     .\" L3308 Germany 7 / 14.3p9 [temp.arg]
 819     .\" Neither local types nor types compounded from them may be used as
 820     .\" template type arguments.
 821     .\" Fixed in Lajoie's pre-London suggested changes to clause 14.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 774     A local type, a type with no linkage, an unnamed type or a type compounded
 775     from any of these types shall not be used as a


Nonmatching lines (File ":original:template"; Line 1152; File ":modified:template";
Line 1200:1204)
1150     .E]
1151     .P
--------------------------------------------------------------------------------
1152     A specialization of a template conversion operator is referenced in

1200     .\" L7171 USA Core3 2.8 N1053 issue 8.11 / 14.2 [temp.names]
1201     .\" Specializations of operator conversion functions are not found by
1202     .\" lookup on their return type, since specializations are not found
1203     .\" by direct lookup.
1204     A specialization of a template conversion operator is declared in
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1153     the same way as a non-template conversion operator that converts to
1154     the same type.


Extra lines in 2nd before 1183 in 1st (File ":original:template"; Line Δ1183; File
":modified:template"; Line 1235:1241)
1181     .N] e
1182     .P
--------------------------------------------------------------------------------
1235     A specialization of a template conversion operator is not found by name
1236     lookup; rather, as with a
1237     .I template-id
1238     (_temp.names_), it is found through the template itself.
1239     A using-declaration in a derived class cannot refer to a specialization
1240     of a template conversion operator in a base class.
1241     .P
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1183     If more than one conversion template can produce the required type,
1184     the partial ordering rules (_temp.func.order_) are used to select the


Extra lines in 2nd before 1366 in 1st (File ":original:template"; Line Δ1366; File
":modified:template"; Line 1425:1432)
1364     implicit instantiation to take place, in every translation unit in which such
a
1365     use occurs.
--------------------------------------------------------------------------------
1425     .\" L7052 USA Core3 1.19 / 14.5.4 [temp.class.spec]
1426     .\" Specialization declarations must be in scope when used in template
instantiations.
1427     When a partial specialization is used within
1428     the instantiation of an exported template, and the unspecialized template
name
1429     is non-dependent in the exported template, a declaration of
1430     the partial specialization must be declared before the definition of the
exported
1431     template, in the translation unit containing that definition.
1432     .P
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1366     Each class template partial specialization is a distinct template and
1367     definitions shall be provided for the members of a template partial


Nonmatching lines (File ":original:template"; Line 1407:1410; File
":modified:template"; Line 1474:1478)
1405     .N]
1406     .P
--------------------------------------------------------------------------------
1407     A class template partial specialization shall be declared in the class or
1408     namespace containing the primary template declaration or in a namespace
1409     of which the class or namespace containing the primary template
1410     declaration is a member.

1474     .\" L6939 USA N1053 Core3 1.4 issue 6.49 and 6.53 item 3 / 14.5.4p4
[temp.class.spec]
1475     .\" Clarify that class template partial specializations in member classes may
```

```
1476     .\" also be declared at namespace scope.
1477     A class template partial specialization may be declared or redeclared in any
1478     namespace scope in which its definition may be defined (_temp.class_ and
_temp.mem_).
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1411     .E[
1412     .Cb


Nonmatching lines (File ":original:template"; Line 1414; File ":modified:template";
Line 1482:1484)
1412     .Cb
1413             template<class T> struct A {
--------------------------------------------------------------------------------
1414                     template<class T2> struct B { };

1482                     class C {
1483                             template<class T2> struct B { };
1484                     };
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1415             };
1416


Nonmatching lines (File ":original:template"; Line 1417:1419; File
":modified:template"; Line 1487:1491)
1415             };
1416
--------------------------------------------------------------------------------
1417             template<class T> template<class T2>
1418                 struct A<T>::B<T2*> { }; // partial specialization of A<T>::B<T2>
1419             A<short>::B<int*> absip;     // uses partial specialization

1487             // partial specialization of A<T>::C::B<T2>
1488             template<class T> template<class T2>
1489                 struct A<T>::C::B<T2*> { };
1490
1491             A<short>::C::B<int*> absip;     // uses partial specialization
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1420     .Ce
1421     .E]


Extra lines in 2nd before 1423 in 1st (File ":original:template"; Line Δ1423; File
":modified:template"; Line 1495:1534)
1421     .E]
1422     .P
--------------------------------------------------------------------------------
1495     .\" L7156 USA Core3 2.7 N1053 issue 6.58 / 14.5.4 [temp.class.spec]
1496     .\" using-declarations refer to primary templates, not class partial
specalizations
1497     Partial specialization declarations themselves are not found by name lookup.
1498     Rather, when the primary template name is used, any previousy declared
partial
1499     specializations of the primary template are also considered.  One consequence
is
1500     that a
1501     .I using-declaration
1502     which refers to a class template does not restrict the set of partial
specializations
1503     which may be found through the
1504     .I using-declaration .
1505     .E[
1506     .Cb
1507             namespace N {
1508                     template<class T1, class T2> class A { }; // primary template
1509             }
1510
1511             using N::A;  // refers to the primary template
1512
1513             namespace N {
1514                     template<class T> class A<T, T*> { }; // partial
specialization
1515             }
1516
1517             A<int,int*> a;  // uses the partial specialization, which is found
through
1518                             // the using declaration which refers to the primary
template
1519     .E]
1520     .\" L7143 USA Core3 2.6 N1053 issue 6.53 / 14.5.4p4 [temp.class.spec]
1521     .\" Clarification of the partial specialization of member class template.
1522     .P
1523     When a member class template of a class template is partially specialized,
1524     the partial specializations are considered declared in all instances
generated
1525     from the enclosing class template.  When primary member class template is
1526     explicitly specialized for a given specialization of the
1527     enclosing class, none of the partial specializations of the original
1528     primary member class template are considered declared in that specialization
of the
1529     enclosing class.
1530     .P
1531     .\" L6947 USA Core3 1.5 N1053 issue 6.51 / 14.5.4p5 / [temp.class.spec]
1532     .\" Restored the less restrictive wording for nontype template arguments
1533     .\" in class template partial specializations.
1534     .\" Fixed in Lajoie's pre-London suggested changes to clause 14.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1423     A non-type argument is non-specialized if it is the name of a non-type
1424     parameter.


Nonmatching lines (File ":original:template"; Line 1444:1445; File
":modified:template"; Line 1556:1559)
1442     .E]
1443     .LI
--------------------------------------------------------------------------------
1444     The type of a specialized nontype argument shall not be dependent on another
1445     type parameter of the specialization.

1556     .\" L6987 USA Core3 1.9 Editorial box 6 / 14.5.4p6 [temp.class.spec]
1557     .\" Fix restrictions on nontype template args in specialization arg lists.
1558     The type of a template parameter corresponding to a specialized nontype
argument
1559     shall not be dependent on a parameter of the specialization.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1446     .E[
1447     .Cb


Extra lines in 2nd before 1514 in 1st (File ":original:template"; Line Δ1514; File
":modified:template"; Line 1628:1631)
1512     .H4 "Partial ordering of class template specializations" temp.class.order
1513     .P
--------------------------------------------------------------------------------
1628     .\" L6955 USA Core3 1.6 N1053 issue 6.52 / 14.5.4.2 [temp.class.order]
1629     .\" Reword the partial ordering rules so that they apply to template
1630     .\" nontype parameters as well as template type parameters.
```

```
1631     .\" Fixed in Lajoie's pre-London suggested changes to clause 14.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1514     For two class template partial specializations,
1515     the first is at least as specialized as the second if, given the following


Extra lines in 2nd before 1635 in 1st (File ":original:template"; Line Δ1635; File
":modified:template"; Line 1753)
1633     .Ce
1634     .E]
----------------------------------------------------------------------------------
1753     .\" L7156 USA Core3 2.7 N1053 issue 6.58 /
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1635     .H3 "Function templates" temp.fct
1636     .P


Extra lines in 2nd before 1700 in 1st (File ":original:template"; Line Δ1700; File
":modified:template"; Line 1819:1822)
1698     .ix "template overload resolution
1699     .ix "function template partial ordering
----------------------------------------------------------------------------------
1819     .\" L6970 USA Core3 1.8 N1053 issue 6.56 / 14.5.5.2 [temp.func.order]
1820     .\" Add additional contexts, e.g. taking the address of a function,
1821     .\" where partial ordering may be used to select among function templates.
1822     .\" Fixed in Lajoie's pre-London suggested changes to clause 14.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1700     If a function template is overloaded,
1701     the use of a function template specialization might be ambiguous because


Extra lines in 2nd before 1893 in 1st (File ":original:template"; Line Δ1893; File
":modified:template"; Line 2016:2018)
1891     .E]
1892     .P
----------------------------------------------------------------------------------
2016     .\" L7106 USA Core3 2.3 core-737 / 14.6p5 [temp.res]
2017     .\" Allow "typename" in a function template return type.
2018     .\" Fixed in Lajoie's pre-London suggested changes to clause 14.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1893     The keyword
1894     .CW typename


Extra lines in 2nd before 1903 in 1st (File ":original:template"; Line Δ1903; File
":modified:template"; Line 2029:2030)
1901     for the definition of a static member of a class template or of a class
nested
1902     within a class template.
----------------------------------------------------------------------------------
2029     .\" L6932 USA Core3 1.3 core-736 parts 2&3 / 14.6p5 [temp.res]
2030     .\" No change, the working paper was correct.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1903     The keyword
1904     .CW typename


Extra lines in 2nd before 1919 in 1st (File ":original:template"; Line Δ1919; File
":modified:template"; Line 2047:2050)
1917     (_temp.dep_) is implicitly assumed to be a type name.
1918     .P
----------------------------------------------------------------------------------
2047     .\" L7016 USA Core3 1.14 core-737 / 14.6p6 [temp.res]
2048     .\" Clarify cases where "typename" is needed even to refer to the
2049     .\" current template.
2050     .\" Fixed in Lajoie's pre-London suggested changes to clause 14.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1919     Within the definition of a class template or within the definition of a
1920     member of a class template, the keyword


Nonmatching lines (File ":original:template"; Line 2081:2084; File
":modified:template"; Line 2213:2219)
2079     .H3 "Locally declared names" temp.local
2080     .P
----------------------------------------------------------------------------------
2081     Within the scope of a class template, the unqualified name of the
2082     template, when not followed by
2083     .CW < ,
2084     is equivalent to the name of the template followed by the

2213     .\" L7180 USA Core3 2.9 Editorial box 8 / 14.6.1p2 [temp.local]
2214     .\" Equivalence of A and A<T> within definition of class template A.
2215     .\" Most of this was in Lajoie's pre-London suggested changes to clause 14.
2216     Within the scope of a class template, when the name of the template is
2217     neither qualified nor followed by
2218     .CW < ,
2219     it is equivalent to the name of the template followed by the
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2085     .I template-parameter s
2086     enclosed in


Extra lines in 1st before 2223 in 2nd (File ":original:template"; Line 2088:2093;
File ":modified:template"; Line Δ2223)
2086     enclosed in
2087     .CW <> .
----------------------------------------------------------------------------------
2088     .N[
2089     the equivalence within the scope of a class template between the name
2090     of the template and the corresponding
2091     .I template-id
2092     does not apply when the name of the template is qualified.
2093     .N] e
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2094     .E[
2095     the constructor for


Nonmatching lines (File ":original:template"; Line 2115:2118; File
":modified:template"; Line 2244:2247)
2113     .E]
2114     .P
----------------------------------------------------------------------------------
2115     Within the scope of a class template specialization, the name of the
2116     specialization, when not followed by
2117     .CW < ,
2118     is equivalent to the name of the specialization

2244     Within the scope of a class template specialization, when the name of the
2245     template is neither qualified nor followed by
2246     .CW < ,
2247     it is equivalent to the name of the template
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2119     followed by the
2120     .I template-argument s
```

```
Extra lines in 2nd before 2732 in 1st (File ":original:template"; Line Δ2732; File
":modified:template"; Line 2861:2863)
2730     .P
2731     .ix "point~of instantiation
--------------------------------------------------------------------------------
2861     .\" L6992 USA Core3 1.10 Editorial box 11 / 14.6.4.1 [temp.point]
2862     .\" Remove an unnecessary rule in the definition of points of instantiation.
2863     .\" Fixed in Lajoie's pre-London suggested changes to clause 14.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2732     For a function template specialization, a member function template
2733     specialization, or a specialization for a member function or static data
member


Extra lines in 2nd before 2793 in 1st (File ":original:template"; Line Δ2793; File
":modified:template"; Line 2925:2926)
2791     definition context or the template instantiation context are found.
2792     .LE
--------------------------------------------------------------------------------
2925     .\" L6996 USA Core3 1.11 Editorial box 12 / 14.6.4.2 [temp.dep.candidate]
2926     .\" The suggestion in the ballot comment was not accepted.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2793     If the call would be ill-formed or would find a better match had the lookup
2794     within the associated namespaces considered all the function declarations
with


Nonmatching lines (File ":original:template"; Line 2798:2816; File
":modified:template"; Line 2932:2937)
2796     not just considered those declarations found in the template definition and
2797     template instantiation contexts, then the program has undefined behavior.
--------------------------------------------------------------------------------
2798     .H4 "Conversions" temp.dep.conv
2799     .P
2800     Any standard conversion sequence (_over.ics.scs_) may be applied to an
2801     argument in a function call that depends on a template argument.
2802     A user-defined conversion sequence (_over.ics.user_) may be applied to an
2803     argument in a function call that depends on a template argument, but
2804     the user-defined conversion in this sequence shall either be a conversion
2805     function that is a member function of the class type of the argument, or
2806     shall be a constructor of the class type that is the target type of the
2807     user-defined conversion sequence.
2808     The user-defined conversion function thus selected shall be found either in
the
2809     template definition context or in the template instantiation context.
2810     .N[
2811     The set of candidate functions is formed first, before
2812     conversions are considered, so the possible conversions do not affect
2813     the set of candidate functions.
2814     .N] e
2815     .H3 "Friend names declared within a class template" temp.inject
2816     .P

2932     .\" L3381 Germany 12 / 14.6.4.3 [temp.dep.conv]
2933     .\" The entire section (one paragraph) was deleted because it is redundant.
2934     .H3 "Friend names declared within a class template" temp.inject
2935     .P
2936     .\" L7003 USA Core3 1.12 public comment 23 / 3.4.2 [basic.lookup.koenig]
2937     .\" and 14.6.5 [temp.inject]   Friend name lookup, as previously agreed.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2817     Friend classes or functions can be declared within a class template.
2818     When a template is instantiated, the names of its friends are treated


Nonmatching lines (File ":original:template"; Line 2822:2824; File
":modified:template"; Line 2943:2947)
2820     instantiation.
2821     .P
--------------------------------------------------------------------------------
2822     The names of friend functions of a class template specialization
2823     are found by the usual lookup rules, including the rules for associated
2824     namespaces (_basic.lookup.koenig_).\*f

2943     As with non-template classes, the names of namespace-scope friend
2944     functions of a class template specialization are not visible during
2945     an ordinary lookup unless explicitly declared at namespace scope
2946     (_class.friend_).  Such names may be found under the rules for associated
2947     classes (_basic.lookup.koenig_).\*f
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2825     .Fs
2826     Friend declarations do not introduce new names into any scope, either


Nonmatching lines (File ":original:template"; Line 2832:2835; File
":modified:template"; Line 2955:2958)
2830     .Cb
2831             template<typename T> class number {
--------------------------------------------------------------------------------
2832                     //...
2833                     friend number<T> gcd(const number<T>& x,
2834                                          const number<T>& y) { ... }
2835                     //...

2955                     number(int);
2956                     //...
2957                     friend number gcd(number& x, number& y) { /* ... */ }
2958                     //...
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2836             };
2837     .Ce


Nonmatching lines (File ":original:template"; Line 2841:2843; File
":modified:template"; Line 2964:2969)
2839             void g()
2840             {
--------------------------------------------------------------------------------
2841                     number<double> a, b;
2842                     //...
2843                     a = gcd(a,b);   // looks inside number<double> for gcd

2964                     number<double> a(3), b(4);
2965                     //...
2966                     a = gcd(a,b);   // finds gcd because number<double> is an
2967                                     // associated class, making gcd visible
2968                                     // in its namespace (global scope)
2969                     b = gcd(3,4);   // ill-formed; gcd is not visible
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2844             }
2845     .Ce


Nonmatching lines (File ":original:template"; Line 2898:2901; File
":modified:template"; Line 3024:3034)
2896     A specialization is a class, function, or class member that is either
2897     instantiated or explicitly specialized (_temp.expl.spec_).
--------------------------------------------------------------------------------
2898     A template that has been used in a way that requires a specialization of its
```

```
2899     definition causes the specialization to be implicitly instantiated unless
2900     a declaration for the explicit specialization appears before the
2901     specialization is used.

3024     .\" L7052 USA Core3 1.19 / 14.5.4 [temp.class.spec]
3025     .\" Specialization declarations must be in scope when used in template
instantiations.
3026     If a template is explicitly specialized then that explicit specialization
shall be
3027     declared before the first use of that specialization that would cause an
3028     implicit instantiation to take place, in every translation unit
3029     in which such a use occurs.
3030     When a specialization for which an explicit specialization exists is used
within
3031     the instantiation of an exported template, and the unspecialized template
name
3032     is non-dependent in the exported template, a declaration of
3033     the explicit specialization must be declared before the definition of the
exported
3034     template, in the translation unit containing that definition.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
2902     .P
2903     No program shall explicitly instantiate any template more than once,


Extra lines in 2nd before 3054 in 1st (File ":original:template"; Line Δ3054; File
":modified:template"; Line 3187:3192)
3052     .Ce
3053     .E]
--------------------------------------------------------------------------------
3187     .\" USA editorial comment, from core3 issue 3.31
3188     .\" Instantiating a class template does not automatically instantiate
3189     .\" a static data member.
3190     .P
3191     The implicit instantiation of a class template does not cause any static data
3192     members of that class to be implicitly instantiated.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
3054     .P
3055     If a function template or a member function template specialization is used
in


Extra lines in 2nd before 3065 in 1st (File ":original:template"; Line Δ3065; File
":modified:template"; Line 3204:3212)
3063     virtual member function of a class template that does not require
3064     specialization.
--------------------------------------------------------------------------------
3204     .\" USA editorial comment, from core3 issue 3.32
3205     .\" A name referenced in a default argument is not "used" unless the default
3206     .\" argument is used.
3207     A reference to a template specialization in a default argument
3208     shall not cause the template to be implicitly instantiated except that a
3209     class template may be instantiated where its complete type is needed to
determine
3210     the correctness of the default argument.  The use of a default argument in a
3211     function call causes specializations in the default argument to be implicitly
3212     instantiated.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
3065     .P
3066     Implicitly instantiated class and function template specializations are
placed


Extra lines in 2nd before 3201 in 1st (File ":original:template"; Line Δ3201; File
":modified:template"; Line 3349:3351)
3199     function (_special_), the program is ill-formed.
3200     .P
--------------------------------------------------------------------------------
3349     .\" L7042 USA Core3 1.17 / 14.7.2p4 [temp.explicit]
3350     .\" The comment asked if an explicit instantiation directive implied
3351     .\" the inclusion model of template compilation.  No, it does not.  No
change.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
3201     The definition of a non-exported function template,
3202     a non-exported member function template,


Extra lines in 2nd before 3745 in 1st (File ":original:template"; Line Δ3745; File
":modified:template"; Line 3896:3905)
3743     argument list for these function templates.
3744     .N] e
--------------------------------------------------------------------------------
3896     .\" L7048 USA Core3 1.18 / 14.8.1 [temp.arg.explicit]
3897     .\" WG21 decided not to address this issue except to document that
3898     .\" argument-dependent lookup does not apply in this context.
3899     .P
3900     .N[
3901     Argument dependent lookup (_basic.lookup.koenig_) is not done when
3902     explicit function template arguments are provided.  If the function
3903     template name is not visible in the current scope, the name must
3904     be explicitly qualified.
3905     .N] e
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
3745     .H3 "Template argument deduction" temp.deduct
3746     .P


Extra lines in 2nd before 3863 in 1st (File ":original:template"; Line Δ3863; File
":modified:template"; Line 4024:4027)
3861     .CW A
3862     via a qualification conversion (_conv.qual_).
--------------------------------------------------------------------------------
4024     .\" L0230 Australia 3 / 14.8.2p4 [temp.deduct]
4025     .\" The comment was a request for an extension: allowing template
4026     .\" type deduction to take user defined conversions into account.
4027     .\" The requested extension was not added.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
3863     .LI
3864     If
```

Clause 15 [except]

```
File #1: :original:except
File #2: :modified:except

Extra lines in 2nd before 254 in 1st (File ":original:except"; Line Δ254; File
":modified:except"; Line 254:265)
 252     statement, that passes control to another handler for
 253     the same exception, so the temporary remains.
--------------------------------------------------------------------------------
 254     .\" L0588 France 11 / 15.1p4 [except.throw]
 255     .\" Clarify the point of destruction of the exception object temporary
 256     When the last handler being executed for the exception exits by
 257     any means other than
 258     .CW throw;
 259     the temporary object is destroyed and the implementation may
 260     deallocate the memory for the temporary object; any such deallocation
 261     is done in an unspecified way.  The destruction occurs immediately
 262     after the destruction of the object declared in the
 263     .I exception-declaration
 264     in the handler.
 265     .P
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 254     If the use of the temporary object can be eliminated without
 255     changing the meaning of the program except for


Extra lines in 2nd before 272 in 1st (File ":original:except"; Line Δ272; File
":modified:except"; Line 284:286)
 270     .ix "rethrow
 271     .ix "reraise
--------------------------------------------------------------------------------
 284     .\" L7009 USA Core3 1.13 public comment 26 / 15.1p5 [except.throw]
 285     .\" Clarify that rethrow does not create a new exception, but
 286     .\" reactivates the old one.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 272     A
 273     .I throw-expression


Nonmatching lines (File ":original:except"; Line 274:275; File ":modified:except";
Line 289:296)
 272     A
 273     .I throw-expression
--------------------------------------------------------------------------------
 274     with no operand rethrows the exception being handled
 275     without copying it.

 289     with no operand rethrows the exception being handled.
 290     The exception is reactivated with the existing temporary;
 291     no new temporary exception object is created.  The exception
 292     is no longer considered to be caught; therefore, the value
 293     of
 294     .CW uncaught_exception()
 295     will again be
 296     .CW true .
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 276     .E[
 277     code that must be executed because of an exception yet cannot


Nonmatching lines (File ":original:except"; Line 324:325; File ":modified:except";
Line 345:351)
 322     of their construction.
 323     .P
--------------------------------------------------------------------------------
 324     An object that is partially constructed will have destructors executed only
 325     for its fully constructed sub-objects.

 345     .\" L0615 France 12 / 15.2p2 [except.ctor]
 346     .\" L7197 USA Core3 2.11 core-769 / 15.2p2 [except.ctor]
 347     .\" Clarify that stack unwinding destroys all fully constructed subobjects.
 348     An object that is partially constructed or partially destroyed will have
 349     destructors executed for all of its fully constructed subobjects,
 350     that is, for subobjects for which the constructor has completed execution
 351     and the destructor has not yet begun execution.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 326     Should a constructor for an element of an automatic array throw an
 327     exception, only the constructed elements of that array will be destroyed.


Extra lines in 2nd before 353 in 1st (File ":original:except"; Line Δ353; File
":modified:except"; Line 379:384)
 351     .I stack
 352     .I unwinding .\(''
--------------------------------------------------------------------------------
 379     .N[
 380     If a destructor called during stack unwinding exits with an exception,
 381     .CW terminate
 382     is called (_except.terminate_).  So destructors should generally catch
 383     exceptions and not let them propagate out of the destructor.
 384     .N]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 353     .H2 "Handling an exception" except.handle
 354     .P


Extra lines in 2nd before 552 in 1st (File ":original:except"; Line Δ552; File
":modified:except"; Line 584:586)
 550     is ill-formed.
 551     .P
--------------------------------------------------------------------------------
 584     .\" L0702 France 13 / 15.3p15 [except.handle]
 585     .\" According to the French representatives, this ballot comment
 586     .\" is not correct.  No action has been taken on this item.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 552     If a return statement appears in a handler of the
 553     .I function-try-block


Extra lines in 2nd before 565 in 1st (File ":original:except"; Line Δ565; File
":modified:except"; Line 600:608)
 563     .I function-try-block
 564     (_stmt.return_).
--------------------------------------------------------------------------------
 600     .\" L0718 France 14 / 15.3p16 [except.handle]
 601     .\" Flowing off the end of a function try block.
 602     Flowing off the end of a
 603     .I function-try-block
 604     is equivalent to a
 605     .CW return
 606     with no value;
 607     this results in undefined behavior in a value-returning function
 608     (_stmt.return_).
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 565     .P
 566     When the
```

```
Nonmatching lines (File ":original:except"; Line 1031; File ":modified:except"; Line
1075:1094)
1029     .H3 "The \f7uncaught_exception()\fP function" except.uncaught
1030     .P
---------------------------------------------------------------------------
1031     See _lib.uncaught_.

1075     .\" USA editorial comment, from core3 issue 3.25
1076     .\" The description of "uncaught_exception" is moved from [lib.uncaught]
1077     .\" to here, and corrected.  (The replacement of the description
1078     .\" in [lib.uncaught] with a cross-reference to [except.uncaught] is an
1079     .\" editorial change.)
1080     The function
1081     .Cb
1082             bool uncaught_exception()
1083     .Ce
1084     returns
1085     .CW true
1086     after completing evaluation of the object to be thrown until completing
1087     the initialization of the
1088     .I exception-declaration
1089     in the matching handler (_lib.uncaught_). This includes stack unwinding.
1090     If the exception is rethrown (_except.throw_),
1091     .CW uncaught_exception()
1092     returns
1093     .CW true
1094     from the point of rethrow until the rethrown exception is caught again.
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1032     .H2 "Exceptions and access" except.access
1033     .P
```