# The Clang AST

A Tutorial by Manuel Klimek

## You'll learn:

1. The basic structure of the Clang AST
2. How to navigate the AST
3. Tools to understand the AST
4. Interfaces to code against the AST (Tooling, AST matchers, etc)

## The Structure of the Clang AST

- rich AST representation
- fully type resolved
- > 100k LOC

## ASTContext

- Keeps information around the AST
  - Identifier Table
  - Source Manager

- Entry point into the AST
  - TranslationUnitDecl* getTranslationUnitDecl()

## Core Classes

- Decl
- Stmt
- Type

## Core Classes

- Decl
  - CXXRecordDecl
  - VarDecl
  - UnresolvedUsingTypenameDecl
- Stmt
- Type

## Core Classes

- Decl
- Stmt
  - CompoundStmt
  - CXXTryStmt
  - BinaryOperator
- Type

## Core Classes

- Decl
- Stmt
- Type
    - PointerType
    - ParenType
    - SubstTemplateTypeParmType

## Glue Classes

- DeclContext
  - inherited by decls that contain other decls
- TemplateArgument
  - accessors for the template argument
- NestedNameSpecifier
- QualType

## Glue Methods

- IfStmt: **getThen**(), **getElse**(), **getCond**()
- CXXRecordDecl: **getDescribedClassTemplate**()
- Type: **getAsCXXRecordDecl**()

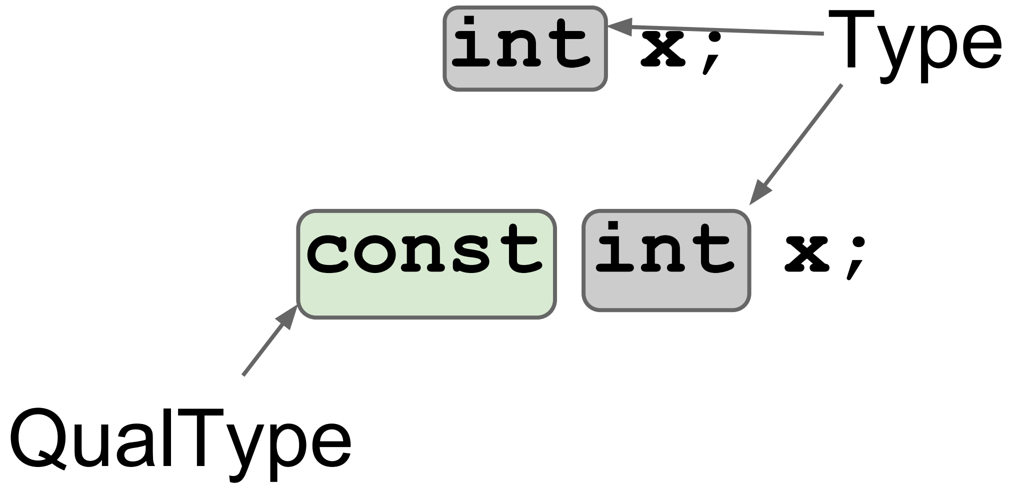
## Types are complicated...

## Types are complicated...

```
int x;

const int x;
```

## Types are complicated...



## Types are complicated...

```
int * const * x;
```

## Types are complicated...

```
class PointerType {

  QualType getPointeeType() const;

};
```

## Types are complicated...

```
int * p;
```

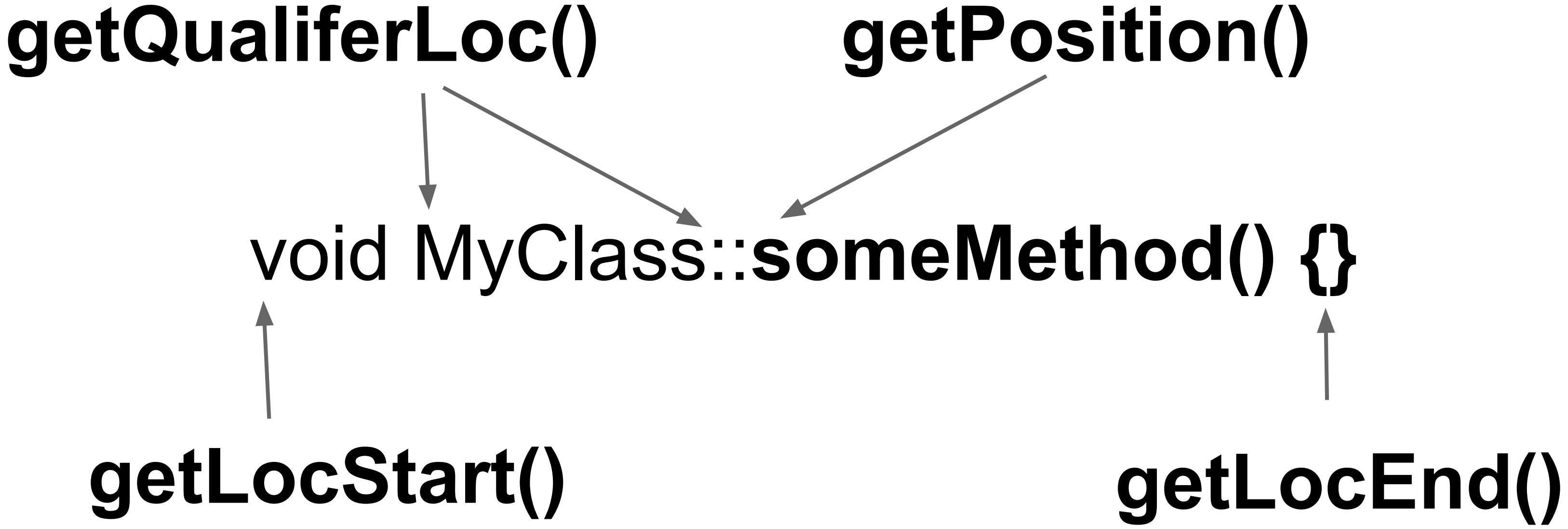
$$\begin{tikzcd}
\text{QualType} \arrow[d] \\
\text{PointerType} \arrow[d] \\
\text{QualType} \arrow[d] \\
\text{BuiltinType}
\end{tikzcd}$$

## Location, Location, Location

class **SourceLocation** { unsigned ID; };

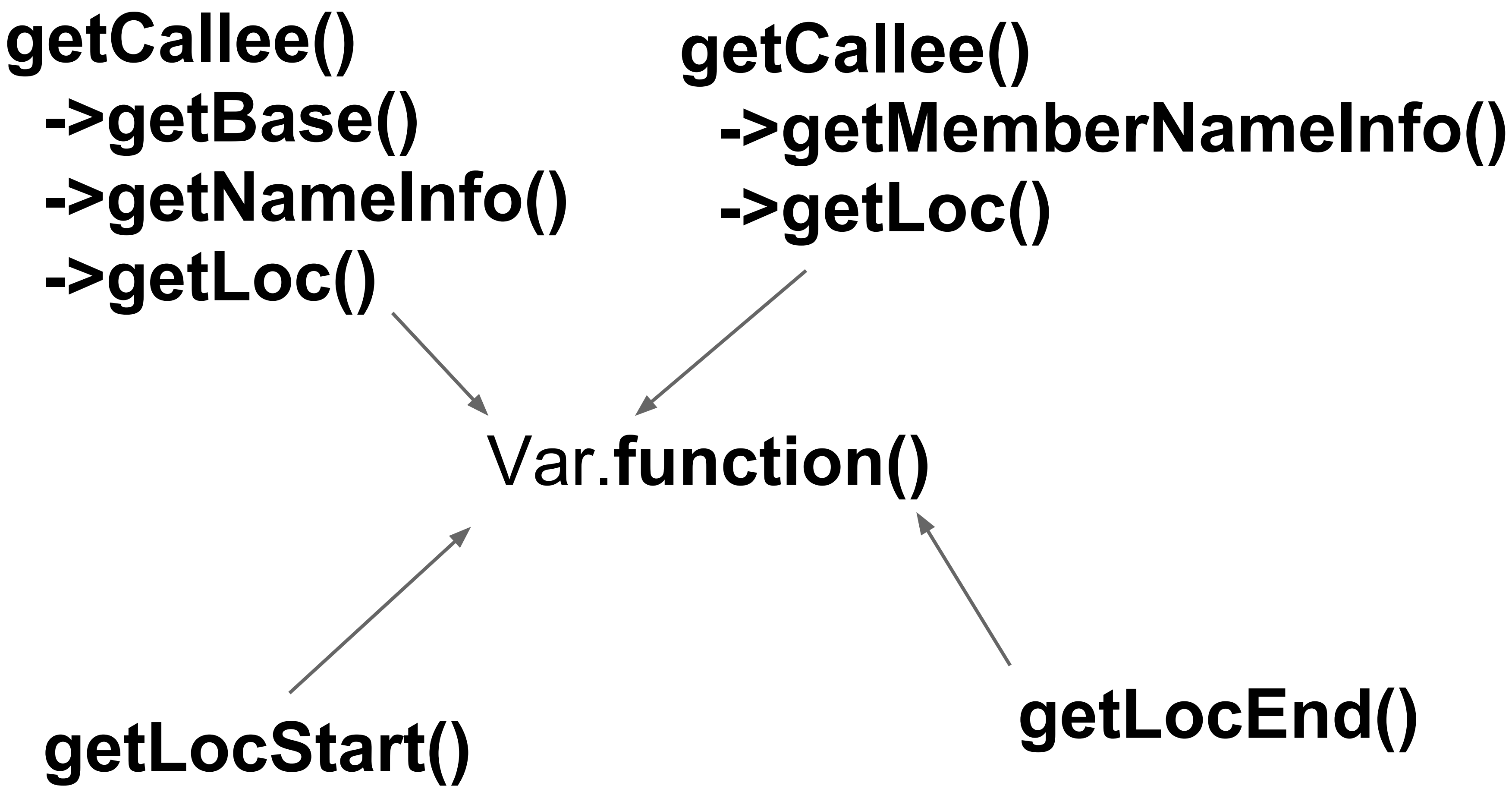
- points to **Tokens**
- managed by **SourceManager**

# Navigating Source: Declarations



```
void MyClass::someMethod() {}
```

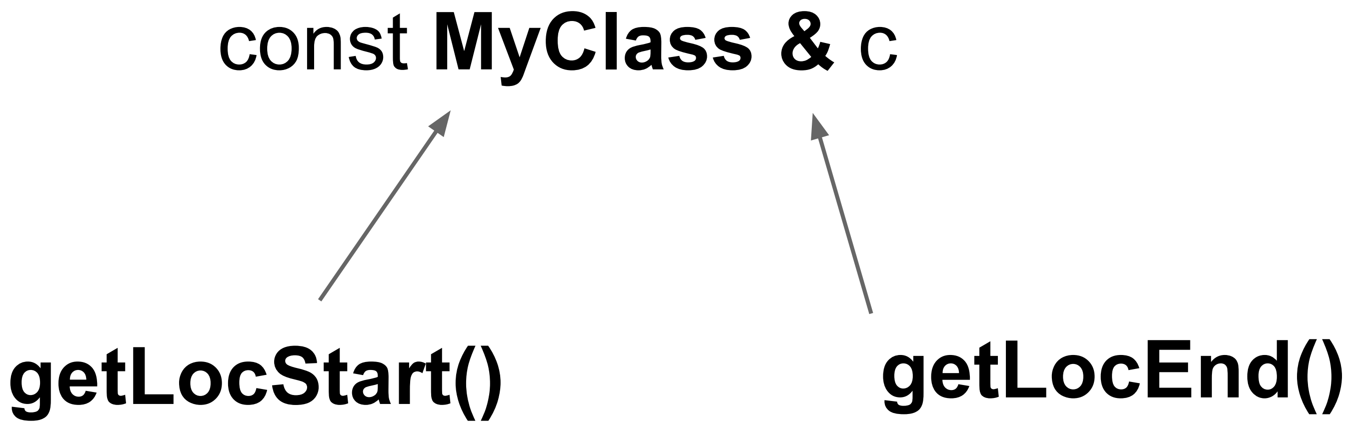
# Navigating Source: Call Expressions



## Navigating Source: Types

$$\begin{tikzcd}
\text{\textbf{MyClass} c;} \arrow[d] & & \text{void f(\textbf{MyClass} c);} \arrow[d] \\
\text{TypeLoc} \arrow[dr] & & \text{TypeLoc} \arrow[dl] \\
& \text{Type: \textbf{MyClass}} &
\end{tikzcd}$$

## Navigating Source: Types



## Navigating Source: Types

**int * p;**

$$\begin{tikzcd}
\text{PointerTypeLoc} \arrow[r] \arrow[dd, "\text{getPointeeLoc()}"] & \text{QualType} \arrow[d] \\
 & \text{PointerType} \arrow[d, "\text{getPointeeType()}"] \\
\text{BuiltinTypeLoc} \arrow[r] & \text{QualType} \arrow[d] \\
 & \text{BuiltinType}
\end{tikzcd}$$

## Getting the Text

Use the **Lexer**!

- makeFileCharRange
- measureTokenLength

## Template tree transformations

- Full AST of **template definition** available
- Full AST for **all instantiations** available
- Nodes are **shared**

## RecursiveASTVisitor

- Trigger on **Types** you care about
- Knows all the connections
- Does not give you context information

## AST Matchers

- Trigger on **Expressions**
- Bind **Context**
- Get all context inside a callback

## Tools!

- clang
  - -ast-dump -ast-dump-filter
  - -ast-list
- clang-check
  - clang + tooling integration

# Example 1: The Real World

```
bool TGParser::AddValue(Record *CurRec, SMLoc Loc,
                        const RecordVal &RV) {
  if (CurRec == 0)
    CurRec = &CurMultiClass->Rec;

  if (RecordVal *ERV = CurRec->getValue(RV.getNameInit())) {
    // The value already exists in the class, treat this as a set.
    if (ERV->setValue(RV.getValue()))
      return Error(Loc, "New definition of '" + RV.getName() + "' of type '"
+
                   RV.getType()->getAsString() + "' is incompatible with " +
                   "previous definition of type '" +
                   ERV->getType()->getAsString() + "'");
  } else {
    CurRec->addValue(RV);
  }
  return false;
}
```

From: llvm/lib/TableGen/TGParser.cpp

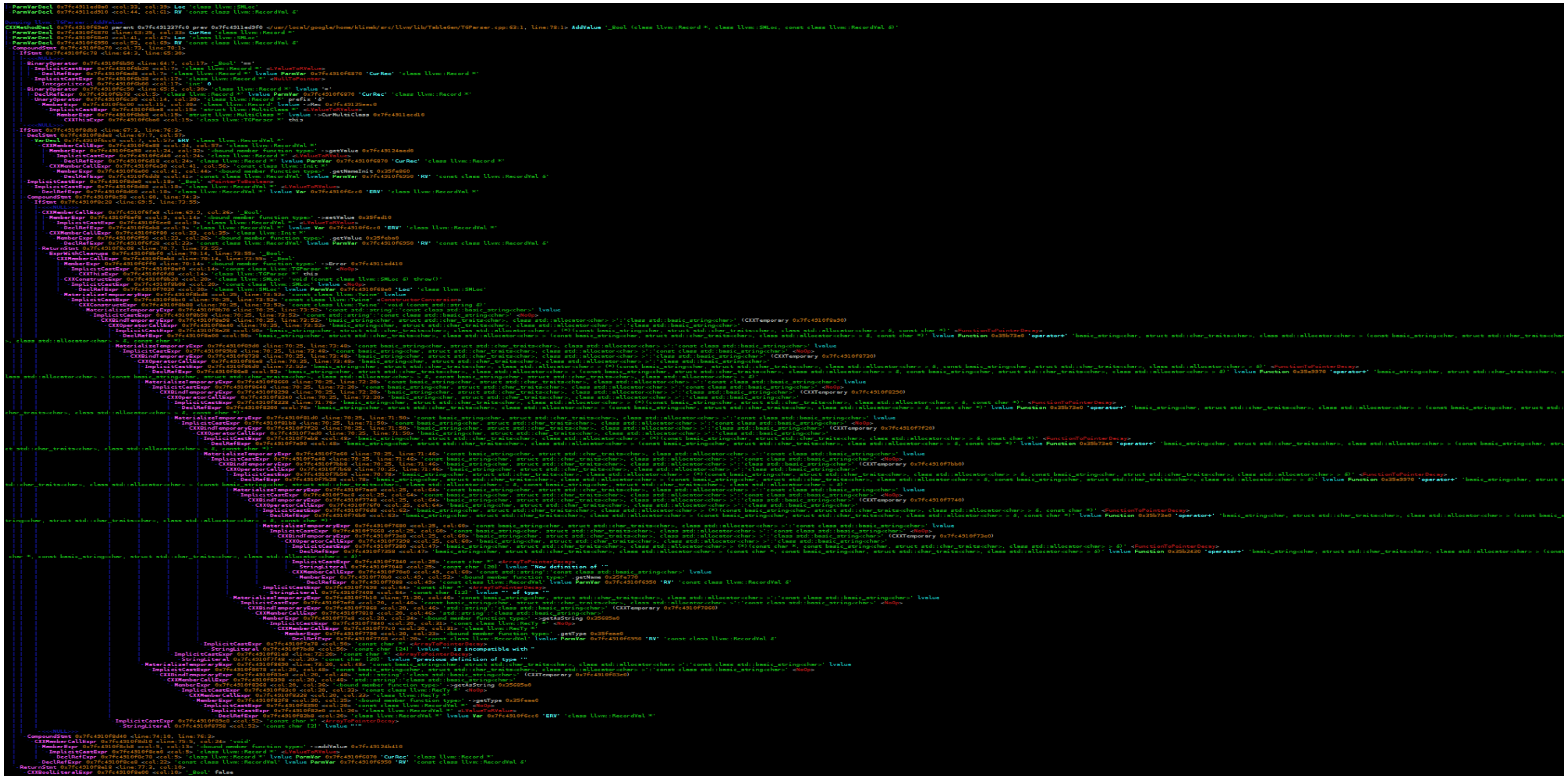
## Example 1: The Real World

Get the AST for **AddValue**

```
$ clang-check -ast-list lib/TableGen/TGParser.cpp \
      |grep AddValue
llvm::TGParser::AddValue
llvm::TGParser::AddValue
```

# Example 1: Dump!

```
$ clang-check -ast-dump \
              -ast-dump-filter=llvm::TGParser::AddValue \
              lib/TableGen/TGParser.cpp
```



```
|-ParmVarDecl 0x7fc4911ed8a0 <col:22, col:29> Loc 'class llvm::SMLoc'
`-ParmVarDecl 0x7fc4911ed910 <col:44, col:61> RV 'const class llvm::RecordVal &'

Dumping llvm::TGParser::AddValue:
CXXMethodDecl 0x7fc4910f69e0 parent 0x7fc491237fc0 prev 0x7fc4911ed9f0 </usr/local/google/home/klimek/src/llvm/lib/TableGen/TGParser.cpp:63:1, line:78:1> AddValue '_Bool (class llvm::Record *, class llvm::SMLoc, const class llvm::RecordVal &)'
|-ParmVarDecl 0x7fc4910f6870 <line:63:25, col:33> CurRec 'class llvm::Record *'
|-ParmVarDecl 0x7fc4910f68e0 <col:41, col:47> Loc 'class llvm::SMLoc'
|-ParmVarDecl 0x7fc4910f6950 <col:52, col:69> RV 'const class llvm::RecordVal &'
`-CompoundStmt 0x7fc4910f8e70 <col:73, line:78:1>
  |-IfStmt 0x7fc4910f6c78 <line:64:3, line:65:30>
  | |-<<<NULL>>>
  | |-BinaryOperator 0x7fc4910f6b50 <line:64:7, col:17> '_Bool' '=='
  | | |-ImplicitCastExpr 0x7fc4910f6b20 <col:7> 'class llvm::Record *' <LValueToRValue>
  | | | `-DeclRefExpr 0x7fc4910f6ad8 <col:7> 'class llvm::Record *' lvalue ParmVar 0x7fc4910f6870 'CurRec' 'class llvm::Record *'
  | | `-ImplicitCastExpr 0x7fc4910f6b38 <col:17> 'class llvm::Record *' <NullToPointer>
  | |   `-IntegerLiteral 0x7fc4910f6b00 <col:17> 'int' 0
  | |-BinaryOperator 0x7fc4910f6c50 <line:65:5, col:30> 'class llvm::Record *' lvalue '='
  | | |-DeclRefExpr 0x7fc4910f6b78 <col:5> 'class llvm::Record *' lvalue ParmVar 0x7fc4910f6870 'CurRec' 'class llvm::Record *'
  | | `-UnaryOperator 0x7fc4910f6c30 <col:14, col:30> 'class llvm::Record *' prefix '&'
  | |   `-MemberExpr 0x7fc4910f6c00 <col:15, col:30> 'class llvm::Record' lvalue ->Rec 0x7fc49125eec0
  | |     `-ImplicitCastExpr 0x7fc4910f6be8 <col:15> 'struct llvm::MultiClass *' <LValueToRValue>
  | |       `-MemberExpr 0x7fc4910f6bb8 <col:15> 'struct llvm::MultiClass *' lvalue ->CurMultiClass 0x7fc4911ecd10
  | |         `-CXXThisExpr 0x7fc4910f6ba0 <col:15> 'class llvm::TGParser *' this
  | `-<<<NULL>>>
  |-IfStmt 0x7fc4910f8db8 <line:67:3, line:76:3>
  | |-DeclStmt 0x7fc4910f8de8 <line:67:7, col:57>
  | | `-VarDecl 0x7fc4910f6cc0 <col:7, col:57> ERV 'class llvm::RecordVal *'
  | |   `-CXXMemberCallExpr 0x7fc4910f6e88 <col:24, col:57> 'class llvm::RecordVal *'
  | |     |-MemberExpr 0x7fc4910f6e58 <col:24, col:32> '<bound member function type>' ->getValue 0x7fc49124aed0
  | |     | `-ImplicitCastExpr 0x7fc4910f6d40 <col:24> 'class llvm::Record *' <LValueToRValue>
  | |     |   `-DeclRefExpr 0x7fc4910f6d18 <col:24> 'class llvm::Record *' lvalue ParmVar 0x7fc4910f6870 'CurRec' 'class llvm::Record *'
  | |     `-CXXMemberCallExpr 0x7fc4910f6e30 <col:41, col:56> 'const class llvm::Init *'
  | |       `-MemberExpr 0x7fc4910f6e00 <col:41, col:44> '<bound member function type>' .getNameInit 0x35fe860
  | |         `-DeclRefExpr 0x7fc4910f6dd8 <col:41> 'const class llvm::RecordVal' lvalue ParmVar 0x7fc4910f6950 'RV' 'const class llvm::RecordVal &'
  | |-ImplicitCastExpr 0x7fc4910f8da0 <col:18> '_Bool' <PointerToBoolean>
  | | `-ImplicitCastExpr 0x7fc4910f8d88 <col:18> 'class llvm::RecordVal *' <LValueToRValue>
  | |   `-DeclRefExpr 0x7fc4910f8d60 <col:18> 'class llvm::RecordVal *' lvalue Var 0x7fc4910f6cc0 'ERV' 'class llvm::RecordVal *'
  | |-CompoundStmt 0x7fc4910f8c58 <col:60, line:74:3>
  | | `-IfStmt 0x7fc4910f8c28 <line:69:5, line:73:55>
  | |   |-<<<NULL>>>
  | |   |-CXXMemberCallExpr 0x7fc4910f6fa8 <line:69:9, col:36> '_Bool'
  | |   | |-MemberExpr 0x7fc4910f6ef8 <col:9, col:14> '<bound member function type>' ->setValue 0x35fed10
  | |   | | `-ImplicitCastExpr 0x7fc4910f6ee0 <col:9> 'class llvm::RecordVal *' <LValueToRValue>
  | |   | |   `-DeclRefExpr 0x7fc4910f6eb8 <col:9> 'class llvm::RecordVal *' lvalue Var 0x7fc4910f6cc0 'ERV' 'class llvm::RecordVal *'
  | |   | `-CXXMemberCallExpr 0x7fc4910f6f80 <col:23, col:35> 'class llvm::Init *'
  | |   |   `-MemberExpr 0x7fc4910f6f50 <col:23, col:26> '<bound member function type>' .getValue 0x35feba0
  | |   |     `-DeclRefExpr 0x7fc4910f6f28 <col:23> 'const class llvm::RecordVal' lvalue ParmVar 0x7fc4910f6950 'RV' 'const class llvm::RecordVal &'
  | |   |-ReturnStmt 0x7fc4910f8c08 <line:70:7, line:73:55>
  | |   | `-ExprWithCleanups 0x7fc4910f8bf0 <line:70:14, line:73:55> '_Bool'
  | |   |   `-CXXMemberCallExpr 0x7fc4910f8ab8 <line:70:14, line:73:55> '_Bool'
  | |   |     |-MemberExpr 0x7fc4910f6ff0 <line:70:14> '<bound member function type>' ->Error 0x7fc4911ed410
  | |   |     | `-ImplicitCastExpr 0x7fc4910f8af0 <col:14> 'const class llvm::TGParser *' <NoOp>
  | |   |     |   `-CXXThisExpr 0x7fc4910f6fd8 <col:14> 'class llvm::TGParser *' this
  | |   |     |-CXXConstructExpr 0x7fc4910f8b20 <col:20> 'class llvm::SMLoc' 'void (const class llvm::SMLoc &) throw()'
  | |   |     | `-ImplicitCastExpr 0x7fc4910f8b08 <col:20> 'const class llvm::SMLoc' lvalue <NoOp>
  | |   |     |   `-DeclRefExpr 0x7fc4910f7020 <col:20> 'class llvm::SMLoc' lvalue ParmVar 0x7fc4910f68e0 'Loc' 'class llvm::SMLoc'
  | |   |     `-MaterializeTemporaryExpr 0x7fc4910f8bd8 <col:25, line:73:52> 'const class llvm::Twine' lvalue
  | |   |       `-ImplicitCastExpr 0x7fc4910f8bc0 <line:70:25, line:73:52> 'const class llvm::Twine' <ConstructorConversion>
  | |   |         `-CXXConstructExpr 0x7fc4910f8b88 <line:70:25, line:73:52> 'const class llvm::Twine' 'void (const std::string &)'
  | |   |           `-MaterializeTemporaryExpr 0x7fc4910f8b70 <line:70:25, line:73:52> 'const std::string':'const class std::basic_string<char>' lvalue
  | |   |             `-ImplicitCastExpr 0x7fc4910f8b58 <line:70:25, line:73:52> 'const std::string':'const class std::basic_string<char>' <NoOp>
  | |   |               `-CXXBindTemporaryExpr 0x7fc4910f8a98 <line:70:25, line:73:52> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>' (CXXTemporary 0x7fc4910f8a90)
  | |   |                 `-CXXOperatorCallExpr 0x7fc4910f8a40 <line:70:25, line:73:52> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>'
  | |   |                   |-ImplicitCastExpr 0x7fc4910f8a28 <col:50> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (*)(const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const char *)' <FunctionToPointerDecay>
  | |   |                   | `-DeclRefExpr 0x7fc4910f8a00 <col:50> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const char *)' lvalue Function 0x35b73e0 'operator+' 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (const basic_string<char, struct std::char_traits<char
>, class std::allocator<char> > &, const char *)'
  | |   |                   |-MaterializeTemporaryExpr 0x7fc4910f89d0 <line:70:25, line:73:48> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' lvalue
  | |   |                   | `-ImplicitCastExpr 0x7fc4910f89b8 <line:70:25, line:73:48> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' <NoOp>
  | |   |                   |   `-CXXBindTemporaryExpr 0x7fc4910f8738 <line:70:25, line:73:48> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>' (CXXTemporary 0x7fc4910f8730)
  | |   |                   |     `-CXXOperatorCallExpr 0x7fc4910f86e8 <line:70:25, line:73:48> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>'
  | |   |                   |       |-ImplicitCastExpr 0x7fc4910f86d0 <line:72:52> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (*)(const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &)' <FunctionToPointerDecay>
  | |   |                   |       | `-DeclRefExpr 0x7fc4910f86a8 <col:52> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &)' lvalue Function 0x35e9970 'operator+' 'basic_string<char, struct std::char_traits<char>, c
lass std::allocator<char> > (const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &)'
  | |   |                   |       |-MaterializeTemporaryExpr 0x7fc4910f8660 <line:70:25, line:72:20> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' lvalue
  | |   |                   |       | `-ImplicitCastExpr 0x7fc4910f8648 <line:70:25, line:72:20> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' <NoOp>
  | |   |                   |       |   `-CXXBindTemporaryExpr 0x7fc4910f8298 <line:70:25, line:72:20> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>' (CXXTemporary 0x7fc4910f8290)
  | |   |                   |       |     `-CXXOperatorCallExpr 0x7fc4910f8240 <line:70:25, line:72:20> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>'
  | |   |                   |       |       |-ImplicitCastExpr 0x7fc4910f8228 <line:71:76> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (*)(const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const char *)' <FunctionToPointerDecay>
  | |   |                   |       |       | `-DeclRefExpr 0x7fc4910f8200 <col:76> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const char *)' lvalue Function 0x35b73e0 'operator+' 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (const basic_string<char, struct std::
char_traits<char>, class std::allocator<char> > &, const char *)'
  | |   |                   |       |       |-MaterializeTemporaryExpr 0x7fc4910f81d0 <line:70:25, line:71:50> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' lvalue
  | |   |                   |       |       | `-ImplicitCastExpr 0x7fc4910f81b8 <line:70:25, line:71:50> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' <NoOp>
  | |   |                   |       |       |   `-CXXBindTemporaryExpr 0x7fc4910f7f28 <line:70:25, line:71:50> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>' (CXXTemporary 0x7fc4910f7f20)
  | |   |                   |       |       |     `-CXXOperatorCallExpr 0x7fc4910f7ed0 <line:70:25, line:71:50> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>'
  | |   |                   |       |       |       |-ImplicitCastExpr 0x7fc4910f7eb8 <col:48> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (*)(const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const char *)' <FunctionToPointerDecay>
  | |   |                   |       |       |       | `-DeclRefExpr 0x7fc4910f7e90 <col:48> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const char *)' lvalue Function 0x35b73e0 'operator+' 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (const basic_string<char, stru
ct std::char_traits<char>, class std::allocator<char> > &, const char *)'
  | |   |                   |       |       |       |-MaterializeTemporaryExpr 0x7fc4910f7e60 <line:70:25, line:71:46> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' lvalue
  | |   |                   |       |       |       | `-ImplicitCastExpr 0x7fc4910f7e48 <line:70:25, line:71:46> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' <NoOp>
  | |   |                   |       |       |       |   `-CXXBindTemporaryExpr 0x7fc4910f7bb8 <line:70:25, line:71:46> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>' (CXXTemporary 0x7fc4910f7bb0)
  | |   |                   |       |       |       |     `-CXXOperatorCallExpr 0x7fc4910f7b68 <line:70:25, line:71:46> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>'
  | |   |                   |       |       |       |       |-ImplicitCastExpr 0x7fc4910f7b50 <line:70:78> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (*)(const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &)' <FunctionToPointerDecay>
  | |   |                   |       |       |       |       | `-DeclRefExpr 0x7fc4910f7b28 <col:78> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &)' lvalue Function 0x35e9970 'operator+' 'basic_string<char, struct s
td::char_traits<char>, class std::allocator<char> > (const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &)'
  | |   |                   |       |       |       |       |-MaterializeTemporaryExpr 0x7fc4910f7ae0 <col:25, col:64> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' lvalue
  | |   |                   |       |       |       |       | `-ImplicitCastExpr 0x7fc4910f7ac8 <col:25, col:64> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' <NoOp>
  | |   |                   |       |       |       |       |   `-CXXBindTemporaryExpr 0x7fc4910f7748 <col:25, col:64> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>' (CXXTemporary 0x7fc4910f7740)
  | |   |                   |       |       |       |       |     `-CXXOperatorCallExpr 0x7fc4910f76f0 <col:25, col:64> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>'
  | |   |                   |       |       |       |       |       |-ImplicitCastExpr 0x7fc4910f76d8 <col:62> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (*)(const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const char *)' <FunctionToPointerDecay>
  | |   |                   |       |       |       |       |       | `-DeclRefExpr 0x7fc4910f76b0 <col:62> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &, const char *)' lvalue Function 0x35b73e0 'operator+' 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (const basic_s
tring<char, struct std::char_traits<char>, class std::allocator<char> > &, const char *)'
  | |   |                   |       |       |       |       |       |-MaterializeTemporaryExpr 0x7fc4910f7680 <col:25, col:60> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' lvalue
  | |   |                   |       |       |       |       |       | `-ImplicitCastExpr 0x7fc4910f7668 <col:25, col:60> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' <NoOp>
  | |   |                   |       |       |       |       |       |   `-CXXBindTemporaryExpr 0x7fc4910f73e8 <col:25, col:60> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>' (CXXTemporary 0x7fc4910f73e0)
  | |   |                   |       |       |       |       |       |     `-CXXOperatorCallExpr 0x7fc4910f7398 <col:25, col:60> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'class std::basic_string<char>'
  | |   |                   |       |       |       |       |       |       |-ImplicitCastExpr 0x7fc4910f7380 <col:47> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (*)(const char *, const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &)' <FunctionToPointerDecay>
  | |   |                   |       |       |       |       |       |       | `-DeclRefExpr 0x7fc4910f7358 <col:47> 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (const char *, const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &)' lvalue Function 0x35b2430 'operator+' 'basic_string<char, struct std::char_traits<char>, class std::allocator<char> > (const
 char *, const basic_string<char, struct std::char_traits<char>, class std::allocator<char> > &)'
  | |   |                   |       |       |       |       |       |       |-ImplicitCastExpr 0x7fc4910f7240 <col:25> 'const char *' <ArrayToPointerDecay>
  | |   |                   |       |       |       |       |       |       | `-StringLiteral 0x7fc4910f7048 <col:25> 'const char [20]' lvalue "New definition of '"
  | |   |                   |       |       |       |       |       |       `-CXXMemberCallExpr 0x7fc4910f70e0 <col:49, col:60> 'const std::string':'const class std::basic_string<char>' lvalue
  | |   |                   |       |       |       |       |       |         `-MemberExpr 0x7fc4910f70b0 <col:49, col:52> '<bound member function type>' .getName 0x35fe770
  | |   |                   |       |       |       |       |       |           `-DeclRefExpr 0x7fc4910f7088 <col:49> 'const class llvm::RecordVal' lvalue ParmVar 0x7fc4910f6950 'RV' 'const class llvm::RecordVal &'
  | |   |                   |       |       |       |       |       `-ImplicitCastExpr 0x7fc4910f7698 <col:64> 'const char *' <ArrayToPointerDecay>
  | |   |                   |       |       |       |       |         `-StringLiteral 0x7fc4910f7408 <col:64> 'const char [12]' lvalue "' of type '"
  | |   |                   |       |       |       |       `-MaterializeTemporaryExpr 0x7fc4910f7b10 <line:71:20, col:46> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' lvalue
  | |   |                   |       |       |       |         `-ImplicitCastExpr 0x7fc4910f7af8 <col:20, col:46> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' <NoOp>
  | |   |                   |       |       |       |           `-CXXBindTemporaryExpr 0x7fc4910f7868 <col:20, col:46> 'std::string':'class std::basic_string<char>' (CXXTemporary 0x7fc4910f7860)
  | |   |                   |       |       |       |             `-CXXMemberCallExpr 0x7fc4910f7818 <col:20, col:46> 'std::string':'class std::basic_string<char>'
  | |   |                   |       |       |       |               `-MemberExpr 0x7fc4910f77e8 <col:20, col:34> '<bound member function type>' ->getAsString 0x35685a0
  | |   |                   |       |       |       |                 `-ImplicitCastExpr 0x7fc4910f7840 <col:20, col:31> 'const class llvm::RecTy *' <NoOp>
  | |   |                   |       |       |       |                   `-CXXMemberCallExpr 0x7fc4910f77c0 <col:20, col:31> 'class llvm::RecTy *'
  | |   |                   |       |       |       |                     `-MemberExpr 0x7fc4910f7790 <col:20, col:22> '<bound member function type>' .getType 0x35feae0
  | |   |                   |       |       |       |                       `-DeclRefExpr 0x7fc4910f7768 <col:20> 'const class llvm::RecordVal' lvalue ParmVar 0x7fc4910f6950 'RV' 'const class llvm::RecordVal &'
  | |   |                   |       |       |       `-ImplicitCastExpr 0x7fc4910f7e78 <col:50> 'const char *' <ArrayToPointerDecay>
  | |   |                   |       |       |         `-StringLiteral 0x7fc4910f7bd8 <col:50> 'const char [24]' lvalue "' is incompatible with "
  | |   |                   |       |       `-ImplicitCastExpr 0x7fc4910f81e8 <line:72:20> 'const char *' <ArrayToPointerDecay>
  | |   |                   |       |         `-StringLiteral 0x7fc4910f7f48 <col:20> 'const char [30]' lvalue "previous definition of type '"
  | |   |                   |       `-MaterializeTemporaryExpr 0x7fc4910f8690 <line:73:20, col:48> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' lvalue
  | |   |                   |         `-ImplicitCastExpr 0x7fc4910f8678 <col:20, col:48> 'const basic_string<char, struct std::char_traits<char>, class std::allocator<char> >':'const class std::basic_string<char>' <NoOp>
  | |   |                   |           `-CXXBindTemporaryExpr 0x7fc4910f83e8 <col:20, col:48> 'std::string':'class std::basic_string<char>' (CXXTemporary 0x7fc4910f83e0)
  | |   |                   |             `-CXXMemberCallExpr 0x7fc4910f8398 <col:20, col:48> 'std::string':'class std::basic_string<char>'
  | |   |                   |               `-MemberExpr 0x7fc4910f8368 <col:20, col:36> '<bound member function type>' ->getAsString 0x35685a0
  | |   |                   |                 `-ImplicitCastExpr 0x7fc4910f83c0 <col:20, col:33> 'const class llvm::RecTy *' <NoOp>
  | |   |                   |                   `-CXXMemberCallExpr 0x7fc4910f8328 <col:20, col:33> 'class llvm::RecTy *'
  | |   |                   |                     `-MemberExpr 0x7fc4910f82f8 <col:20, col:25> '<bound member function type>' ->getType 0x35feae0
  | |   |                   |                       `-ImplicitCastExpr 0x7fc4910f8250 <col:20> 'const class llvm::RecordVal *' <NoOp>
  | |   |                   |                         `-ImplicitCastExpr 0x7fc4910f82e0 <col:20> 'class llvm::RecordVal *' <LValueToRValue>
  | |   |                   |                           `-DeclRefExpr 0x7fc4910f82b8 <col:20> 'class llvm::RecordVal *' lvalue Var 0x7fc4910f6cc0 'ERV' 'class llvm::RecordVal *'
  | |   |                   `-ImplicitCastExpr 0x7fc4910f89e8 <col:52> 'const char *' <ArrayToPointerDecay>
  | |   |                     `-StringLiteral 0x7fc4910f8758 <col:52> 'const char [2]' lvalue "'"
  | |   `-<<<NULL>>>
  | `-CompoundStmt 0x7fc4910f8d40 <line:74:10, line:76:3>
  |   `-CXXMemberCallExpr 0x7fc4910f8d10 <line:75:5, col:24> 'void'
  |     |-MemberExpr 0x7fc4910f8cb8 <col:5, col:13> '<bound member function type>' ->addValue 0x7fc49124b410
  |     | `-ImplicitCastExpr 0x7fc4910f8ca0 <col:5> 'class llvm::Record *' <LValueToRValue>
  |     |   `-DeclRefExpr 0x7fc4910f8c78 <col:5> 'class llvm::Record *' lvalue ParmVar 0x7fc4910f6870 'CurRec' 'class llvm::Record *'
  |     `-DeclRefExpr 0x7fc4910f8ce8 <col:22> 'const class llvm::RecordVal' lvalue ParmVar 0x7fc4910f6950 'RV' 'const class llvm::RecordVal &'
  `-ReturnStmt 0x7fc4910f8e18 <line:77:3, col:10>
    `-CXXBoolLiteralExpr 0x7fc4910f8e00 <col:10> '_Bool' false
```

# Example 1: Dump Details

```
<...>
|-ReturnStmt 0x7f9047a23c28 <line:70:7, line:73:55>
| `-ExprWithCleanups 0x7f9047a23c10 <line:70:14, line:73:55> '_Bool'
|   `-CXXMemberCallExpr 0x7f9047a23ad8 <line:70:14, line:73:55> '_Bool'
|     |-MemberExpr 0x7f9047a21ff0 <line:70:14> '<bound member function type>' ->Error 0x7f9047b18410
|     | `-ImplicitCastExpr 0x7f9047a23b10 <col:14> 'const class llvm::TGParser *' <NoOp>
|     |   `-CXXThisExpr 0x7f9047a21fd8 <col:14> 'class llvm::TGParser *' this
|     |-CXXConstructExpr 0x7f9047a23b40 <col:20> 'class llvm::SMLoc' 'void (const class llvm::SMLoc &) throw()'
|     | `-ImplicitCastExpr 0x7f9047a23b28 <col:20> 'const class llvm::SMLoc' lvalue <NoOp>
|     |   `-DeclRefExpr 0x7f9047a22020 <col:20> 'class llvm::SMLoc' lvalue ParmVar 0x7f9047a218e0 'Loc' 'class llvm::SMLoc'
|     `-MaterializeTemporaryExpr 0x7f9047a23bf8 <col:25, line:73:52> 'const class llvm::Twine' lvalue
|       `-ImplicitCastExpr 0x7f9047a23be0 <line:70:25, line:73:52> 'const class llvm::Twine' <ConstructorConversion>
|         `-CXXConstructExpr 0x7f9047a23ba8 <line:70:25, line:73:52> 'const class llvm::Twine' 'void (const std::string &)'
<...>
```

## Example 2: std::string Arguments

```
#include <string>

void f(const std::string& s);

void StdStringArgumentCall(
          const std::string& s) {
  f(s.c_str());
}
```

# Example 2: Dump!

```
$ clang-check StdStringArgs.cc -ast-dump -ast-dump-filter=StdStringA --

Dumping StdStringArgumentCall:
FunctionDecl
|-ParmVarDecl
`-CompoundStmt
  `-ExprWithCleanups
    `-CallExpr
      |-ImplicitCastExpr <FunctionToPointerDecay>
      | `-DeclRefExpr 'f' 'void (const std::string &)'
      `-MaterializeTemporaryExpr
        `-CXXBindTemporaryExpr
          `-CXXConstructExpr 'void (const char *, const class std::allocator<char> &)'
            |-CXXMemberCallExpr 'const char *'
            | `-MemberExpr .c_str
            |   `-DeclRefExpr 's' 'const std::string &'
            `-CXXDefaultArgExpr 'const class std::allocator<char>'
```

## Example 2: Dump!

```
$ clang-check StdStringArgs.cc -ast-dump -ast-dump-filter=StdStringA --

Dumping StdStringArgumentCall:
FunctionDecl
|-ParmVarDecl
`-CompoundStmt
  `-ExprWithCleanups
    `-CallExpr
      |-ImplicitCastExpr <FunctionToPointerDecay>
      | `-DeclRefExpr 'f' 'void (const std::string &)'
      `-MaterializeTemporaryExpr
        `-CXXBindTemporaryExpr
          `-CXXConstructExpr 'void (const char *, const class std::allocator<char> &)'
            |-CXXMemberCallExpr 'const char *'
            | `-MemberExpr .c_str                          s.c_str()
            |   `-DeclRefExpr 's' 'const std::string &'
            `-CXXDefaultArgExpr 'const class std::allocator<char>'
```

## Example 2: Dump!

```
$ clang-check StdStringArgs.cc -ast-dump -ast-dump-filter=StdStringA --

Dumping StdStringArgumentCall:
FunctionDecl
|-ParmVarDecl
`-CompoundStmt
  `-ExprWithCleanups
    `-CallExpr
      |-ImplicitCastExpr <FunctionToPointerDecay>
      | `-DeclRefExpr 'f' 'void (const std::string &)'
      `-MaterializeTemporaryExpr
        `-CXXBindTemporaryExpr
          `-CXXConstructExpr 'void (const char *, const class std::allocator<char> &)'
            |-CXXMemberCallExpr 'const char *'
            | `-MemberExpr .c_str
            |   `-DeclRefExpr 's' 'const std::string &'
            `-CXXDefaultArgExpr 'const class std::allocator<char>'
```

**string(s.c_str())**

# Example 2: Dump!

```
$ clang-check StdStringArgs.cc -ast-dump -ast-dump-filter=StdStringA --

Dumping StdStringArgumentCall:
FunctionDecl
|-ParmVarDecl
`-CompoundStmt
  `-ExprWithCleanups
    `-CallExpr
      |-ImplicitCastExpr <FunctionToPointerDecay>
      | `-DeclRefExpr 'f' 'void (const std::string &)'
      `-MaterializeTemporaryExpr
        `-CXXBindTemporaryExpr
          `-CXXConstructExpr 'void (const char *, const class std::allocator<char> &)'
            |-CXXMemberCallExpr 'const char *'
            | `-MemberExpr .c_str
            |   `-DeclRefExpr 's' 'const std::string &'
            `-CXXDefaultArgExpr 'const class std::allocator<char>'
```

**f(s.c_str())**

## Example 2: std::string Arguments

```
#include <string>

void f(const std::string& s);

void StdStringArgumentCall(
          const std::string& s) {
  f(s.c_str());
}
```

## Example 2: std::string Arguments

```
#include <string>

void f(const std::string& s);

void StdStringArgumentCall(
          const std::string& s) {
  f(std::string(s.c_str()));
}
```

## Example 2: Dump!

```
Dumping StdStringArgumentCall:
FunctionDecl
|-ParmVarDecl
`-CompoundStmt
  `-ExprWithCleanups
    `-CallExpr
      |-ImplicitCastExpr <FunctionToPointerDecay>
      | `-DeclRefExpr 'f' 'void (const std::string &)'
      `-MaterializeTemporaryExpr
        `-CXXBindTemporaryExpr
          `-CXXConstructExpr
            |-CXXMemberCallExpr 'const char *'
            | `-MemberExpr .c_str
            |   `-DeclRefExpr 's' 'const std::string &'
            `-CXXDefaultArgExpr 'const class std::allocator<char>'
```

# Example 2: Dump!

```
Dumping StdStringArgumentCall:
FunctionDecl
|-ParmVarDecl
`-CompoundStmt
  `-ExprWithCleanups
    `-CallExpr
      |-ImplicitCastExpr <FunctionToPointerDecay>
      | `-DeclRefExpr 'f' 'void (const std::string &)'
      `-MaterializeTemporaryExpr
        `-ImplicitCastExpr <NoOp>
          `-CXXFunctionalCastExpr to std::string <ConstructorConversion>
            `-CXXBindTemporaryExpr
              `-CXXConstructExpr
                |-CXXMemberCallExpr 'const char *'
                | `-MemberExpr .c_str
                |   `-DeclRefExpr 's' 'const std::string &'
                `-CXXDefaultArgExpr 'const class std::allocator<char>'
```

# Example 2: Dump!

```
Dumping StdStringArgumentCall:
FunctionDecl
|-ParmVarDecl
`-CompoundStmt
  `-ExprWithCleanups
    `-CallExpr
      |-ImplicitCastExpr <FunctionToPointerDecay>
      | `-DeclRefExpr 'f' 'void (const std::string &)'
      `-MaterializeTemporaryExpr
        `-ImplicitCastExpr <NoOp>
          `-CXXFunctionalCastExpr to std::string <ConstructorConversion>
            `-CXXBindTemporaryExpr
              `-CXXConstructExpr
                |-CXXMemberCallExpr 'const char *'
                | `-MemberExpr .c_str
                |   `-DeclRefExpr 's' 'const std::string &'
                `-CXXDefaultArgExpr 'const class std::allocator<char>'
```

# Getting Real

```
#include "clang/ASTMatchers/ASTMatchers.h"
#include "clang/ASTMatchers/ASTMatchFinder.h"
#include "clang/Tooling/Tooling.h"
#include "gtest/gtest.h"

using namespace llvm;
using namespace clang;
using namespace clang::tooling;
using namespace clang::ast_matchers;

class DumpCallback : public MatchFinder::MatchCallback {
  virtual void run(const MatchFinder::MatchResult &Result) {
    llvm::errs() << "---\n";
    Result.Nodes.getNodeAs<CXXRecordDecl>("x")->dump();
  }
};

TEST(DumpCodeSample, Dumps) {
  DumpCallback Callback;
  MatchFinder Finder;
  Finder.addMatcher(recordDecl().bind("x"), &Callback);
  OwningPtr<FrontendActionFactory> Factory(newFrontendActionFactory(&Finder));
  EXPECT_TRUE(clang::tooling::runToolOnCode(Factory->create(), "class X {};"));
}
```

## Getting Real

```
class DumpCallback : public MatchFinder::MatchCallback {
  virtual void run(const MatchFinder::MatchResult &Result) {
    llvm::errs() << "---\n";

    const CXXRecordDecl *D = Result.Nodes.getNodeAs<CXXRecordDecl>("x");
    if (const clang::ClassTemplateSpecializationDecl *TS =
            dyn_cast<clang::ClassTemplateSpecializationDecl>(D)) {
      TS->getLocation().dump(*Result.SourceManager);
      llvm::errs() << "\n";
    }
  }
};

"template <typename T> class X {}; X<int> y;"
```

## Links

http://clang.llvm.org/docs/Tooling.html
http://clang.llvm.org/docs/IntroductionToTheClangAST.html
http://clang.llvm.org/docs/RAVFrontendAction.html
http://clang.llvm.org/docs/LibTooling.html
http://clang.llvm.org/docs/LibASTMatchers.html