How to use llvm-debuginfo-analyzer tool

Carlos Alberto Enciso

How to use llvm-debuginfo-analyzer

A. Introduction
   1. Common problems with debug information
   2. LLVM and debug information

B. llvm-debuginfo-analyzer
   1. Introduction
   2. Print options
   3. Select options
   4. Compare options

C. Future work

A. Introduction

1. Common problems with debug information

- Does the debug information represent the original source
  - Which variables are dropped due to optimization
  - Why I cannot stop at a particular line
  - Which lines are associated to a specific code range
  - Size changes due to toolchain features

- Semantic differences in the generated debug information
  - By different toolchain versions (same platform)
    - Clang 1.0.1 and 1.0.2
    - Clang and GCC
  - By same or different toolchain versions (different platforms)
    - Clang (Windows) and Clang (Linux)
    - Clang (Linux) and MSVC (Windows)

Debugger - Incorrect lexical scope for typedef ‘TYPE’ (PR44229)

```
 1 int main() {
 2   typedef unsigned TYPE;
 3   int Result = 0;
 4   {
 5     typedef float TYPE;
 6     TYPE Var_1 = 123.45;
 7     Result += Var_1;
 8   }
 9   TYPE Var_2 = 123;
10   return Result + Var_2;
11 }
```

GDB debug session (right)

Line 3: ‘TYPE’ definition
Expected: unsigned
Reported: float

Line 9: ‘TYPE’ definition
Expected: unsigned
Reported: float

Example use case

```
Reading symbols from test.out...done.
(gdb) b 3
(gdb) b 6
(gdb) b 9
(gdb) run

Breakpoint 1, main () at test.cpp:3
3     int Result = 0;
(gdb) whatis TYPE
type = float

Breakpoint 2, main () at test.cpp:6
6       TYPE Var_1 = 123.45;
(gdb) whatis TYPE
type = float

Breakpoint 3, main () at test.cpp:9
9     TYPE Var_2 = 123;
(gdb) whatis TYPE
type = float
```

At line 3, the typedef 'TYPE' should be defined as 'unsigned’.

Due to a bug in the compiler, GDB incorrectly reports the definition as 'float’.

GDB debug session

Debugger - Incorrect lexical scope for typedef ‘TYPE’ (PR44229)

```
 1 int main() {
 2   typedef unsigned TYPE;
 3   int Result = 0;
 4   {
 5     typedef float TYPE;
 6     TYPE Var_1 = 123.45;
 7     Result += Var_1;
 8   }
 9   TYPE Var_2 = 123;
10   return Result + Var_2;
11 }
```

GDB debug session (right)

Line 3: ‘TYPE’ definition
Expected: unsigned
Reported: float

Line 9: ‘TYPE’ definition
Expected: unsigned
Reported: float

```
Reading symbols from test.out...done.
(gdb) b 3
(gdb) b 6
(gdb) b 9
(gdb) run

Breakpoint 1, main () at test.cpp:3
3     int Result = 0;
(gdb) whatis TYPE
type = float

Breakpoint 2, main () at test.cpp:6
6       TYPE Var_1 = 123.45;
(gdb) whatis TYPE
type = float

Breakpoint 3, main () at test.cpp:9
9     TYPE Var_2 = 123;
(gdb) whatis TYPE
type = float

At line 3, the typedef 'TYPE' should be defined as 'unsigned’.

Due to a bug in the compiler, GDB incorrectly reports the definition as
'float’.
```

Example use case

GDB debug session

A. Introduction

2. LLVM and debug information

- Several debug information formats
  - DWARF, CodeView
- Different binary file formats
  - ELF, COFF, PDB, Mach-O
- Different tools to print the debug information
  - llvm-dwarfdump, llvm-pdbutil, llvm-readelf
  - They use a close representation to the internal formats
  - Requires good knowledge of the format's specifications
- Understanding mappings between source code and debug information can be complex
- It is a problem commonly encountered when triaging LLVM’s debug information issues

DWARF vs CodeView Debug Information

```
 1 int main() {
 2   typedef unsigned TYPE;
 3   int Result = 0;
 4   {
 5     typedef float TYPE;
 6     TYPE Var_1 = 123.45;
 7     Result += Var_1;
 8   }
 9   TYPE Var_2 = 123;
10   return Result + Var_2;
11 }
```

Example use case

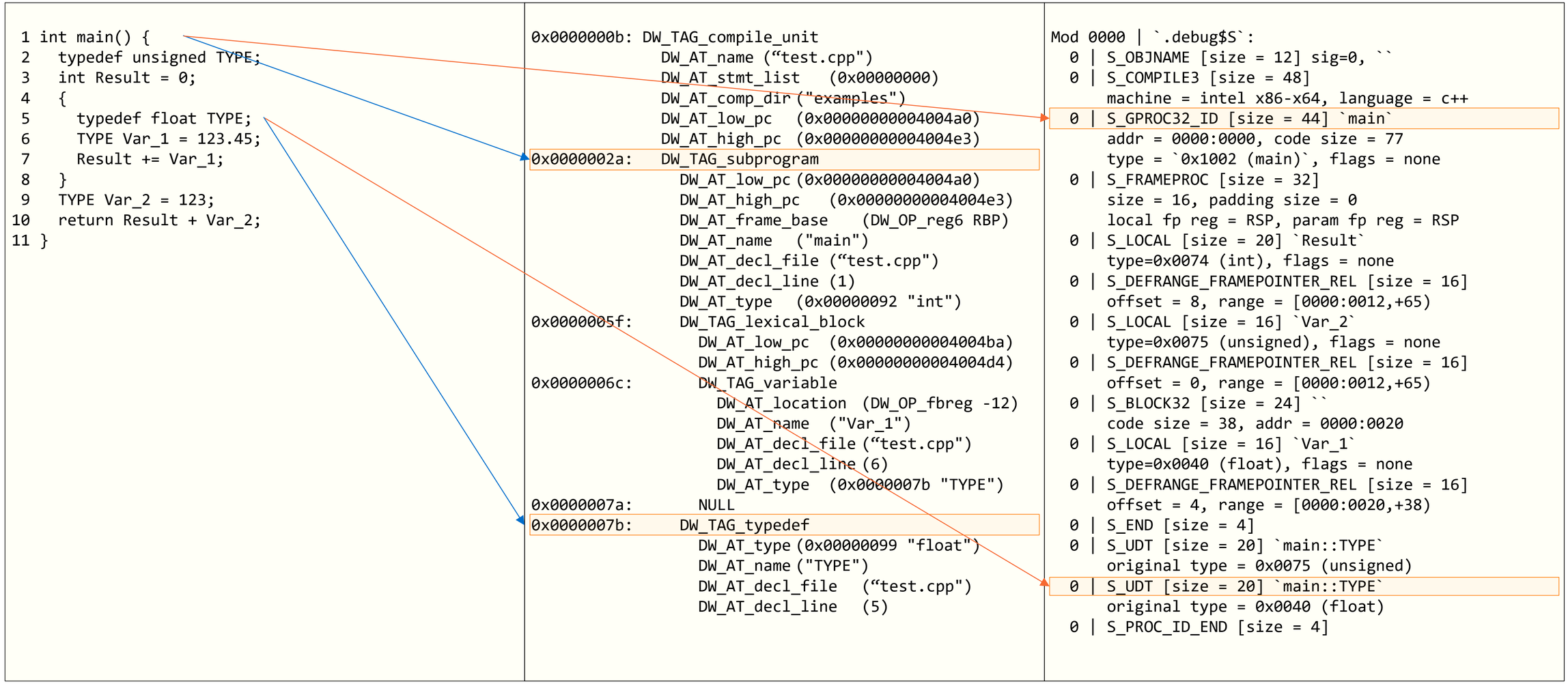
```
0x0000000b: DW_TAG_compile_unit
              DW_AT_name ("test.cpp")
              DW_AT_stmt_list   (0x00000000)
              DW_AT_comp_dir ("examples")
              DW_AT_low_pc   (0x00000000004004a0)
              DW_AT_high_pc  (0x00000000004004e3)
0x0000002a:   DW_TAG_subprogram
                DW_AT_low_pc (0x00000000004004a0)
                DW_AT_high_pc   (0x00000000004004e3)
                DW_AT_frame_base    (DW_OP_reg6 RBP)
                DW_AT_name   ("main")
                DW_AT_decl_file ("test.cpp")
                DW_AT_decl_line (1)
                DW_AT_type   (0x00000092 "int")
0x0000005f:     DW_TAG_lexical_block
                  DW_AT_low_pc  (0x00000000004004ba)
                  DW_AT_high_pc (0x00000000004004d4)
0x0000006c:       DW_TAG_variable
                    DW_AT_location  (DW_OP_fbreg -12)
                    DW_AT_name  ("Var_1")
                    DW_AT_decl_file ("test.cpp")
                    DW_AT_decl_line (6)
                    DW_AT_type  (0x0000007b "TYPE")
0x0000007a:       NULL
0x0000007b:     DW_TAG_typedef
                  DW_AT_type (0x00000099 "float")
                  DW_AT_name ("TYPE")
                  DW_AT_decl_file   ("test.cpp")
                  DW_AT_decl_line   (5)
```

DWARF debug information dump

```
Mod 0000 | `.debug$S`:
  0 | S_OBJNAME [size = 12] sig=0, ``
  0 | S_COMPILE3 [size = 48]
      machine = intel x86-x64, language = c++
  0 | S_GPROC32_ID [size = 44] `main`
      addr = 0000:0000, code size = 77
      type = `0x1002 (main)`, flags = none
  0 | S_FRAMEPROC [size = 32]
      size = 16, padding size = 0
      local fp reg = RSP, param fp reg = RSP
  0 | S_LOCAL [size = 20] `Result`
      type=0x0074 (int), flags = none
  0 | S_DEFRANGE_FRAMEPOINTER_REL [size = 16]
      offset = 8, range = [0000:0012,+65)
  0 | S_LOCAL [size = 16] `Var_2`
      type=0x0075 (unsigned), flags = none
  0 | S_DEFRANGE_FRAMEPOINTER_REL [size = 16]
      offset = 0, range = [0000:0012,+65)
  0 | S_BLOCK32 [size = 24] ``
      code size = 38, addr = 0000:0020
  0 | S_LOCAL [size = 16] `Var_1`
      type=0x0040 (float), flags = none
  0 | S_DEFRANGE_FRAMEPOINTER_REL [size = 16]
      offset = 4, range = [0000:0020,+38)
  0 | S_END [size = 4]
  0 | S_UDT [size = 20] `main::TYPE`
      original type = 0x0075 (unsigned)
  0 | S_UDT [size = 20] `main::TYPE`
      original type = 0x0040 (float)
  0 | S_PROC_ID_END [size = 4]
```

CodeView debug information dump

DWARF vs CodeView Debug Information



```
 1 int main() {
 2   typedef unsigned TYPE;
 3   int Result = 0;
 4   {
 5     typedef float TYPE;
 6     TYPE Var_1 = 123.45;
 7     Result += Var_1;
 8   }
 9   TYPE Var_2 = 123;
10   return Result + Var_2;
11 }
```

```
0x0000000b: DW_TAG_compile_unit
              DW_AT_name ("test.cpp")
              DW_AT_stmt_list   (0x00000000)
              DW_AT_comp_dir ("examples")
              DW_AT_low_pc   (0x00000000004004a0)
              DW_AT_high_pc  (0x00000000004004e3)
0x0000002a:   DW_TAG_subprogram
                DW_AT_low_pc (0x00000000004004a0)
                DW_AT_high_pc   (0x00000000004004e3)
                DW_AT_frame_base    (DW_OP_reg6 RBP)
                DW_AT_name   ("main")
                DW_AT_decl_file ("test.cpp")
                DW_AT_decl_line (1)
                DW_AT_type   (0x00000092 "int")
0x0000005f:     DW_TAG_lexical_block
                  DW_AT_low_pc  (0x00000000004004ba)
                  DW_AT_high_pc (0x00000000004004d4)
0x0000006c:       DW_TAG_variable
                    DW_AT_location  (DW_OP_fbreg -12)
                    DW_AT_name  ("Var_1")
                    DW_AT_decl_file ("test.cpp")
                    DW_AT_decl_line (6)
                    DW_AT_type  (0x0000007b "TYPE")
0x0000007a:       NULL
0x0000007b:     DW_TAG_typedef
                  DW_AT_type (0x00000099 "float")
                  DW_AT_name ("TYPE")
                  DW_AT_decl_file   ("test.cpp")
                  DW_AT_decl_line   (5)
```

```
Mod 0000 | `.debug$S`:
  0 | S_OBJNAME [size = 12] sig=0, ``
  0 | S_COMPILE3 [size = 48]
      machine = intel x86-x64, language = c++
  0 | S_GPROC32_ID [size = 44] `main`
      addr = 0000:0000, code size = 77
      type = `0x1002 (main)`, flags = none
  0 | S_FRAMEPROC [size = 32]
      size = 16, padding size = 0
      local fp reg = RSP, param fp reg = RSP
  0 | S_LOCAL [size = 20] `Result`
      type=0x0074 (int), flags = none
  0 | S_DEFRANGE_FRAMEPOINTER_REL [size = 16]
      offset = 8, range = [0000:0012,+65)
  0 | S_LOCAL [size = 16] `Var_2`
      type=0x0075 (unsigned), flags = none
  0 | S_DEFRANGE_FRAMEPOINTER_REL [size = 16]
      offset = 0, range = [0000:0012,+65)
  0 | S_BLOCK32 [size = 24] ``
      code size = 38, addr = 0000:0020
  0 | S_LOCAL [size = 16] `Var_1`
      type=0x0040 (float), flags = none
  0 | S_DEFRANGE_FRAMEPOINTER_REL [size = 16]
      offset = 4, range = [0000:0020,+38)
  0 | S_END [size = 4]
  0 | S_UDT [size = 20] `main::TYPE`
      original type = 0x0075 (unsigned)
  0 | S_UDT [size = 20] `main::TYPE`
      original type = 0x0040 (float)
  0 | S_PROC_ID_END [size = 4]
```

Example use case | **DWARF debug information dump** | **CodeView debug information dump**

B. llvm-debuginfo-analyzer

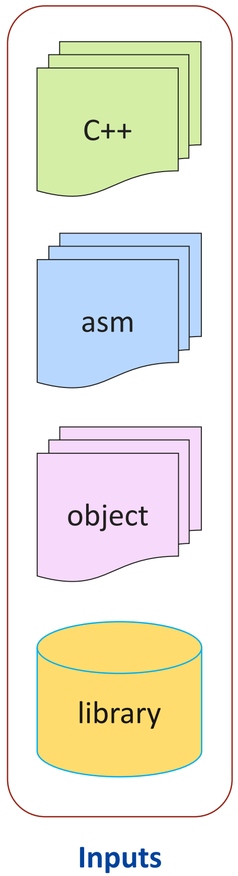
1. Introduction

- Command line tool that processes the debugging information
  - Supported debug information formats: DWARF, CodeView
  - Supported binary file formats: ELF, COFF, PDB, Mach-O

- Produces a logical view, which is a high-level representation of the debug information
  - Composed of logical elements: scopes, types, symbols, and lines
  - Can display additional attributes: variable coverage factor, lexical block level, template argument encoding, etc.

- Key features
  - Uniform logical view regardless of the debug information and binary file formats
  - Logical lines associated to their logical scopes
  - Criteria used to select which logical elements to include in the logical view
  - Find semantic differences by comparing the logical views

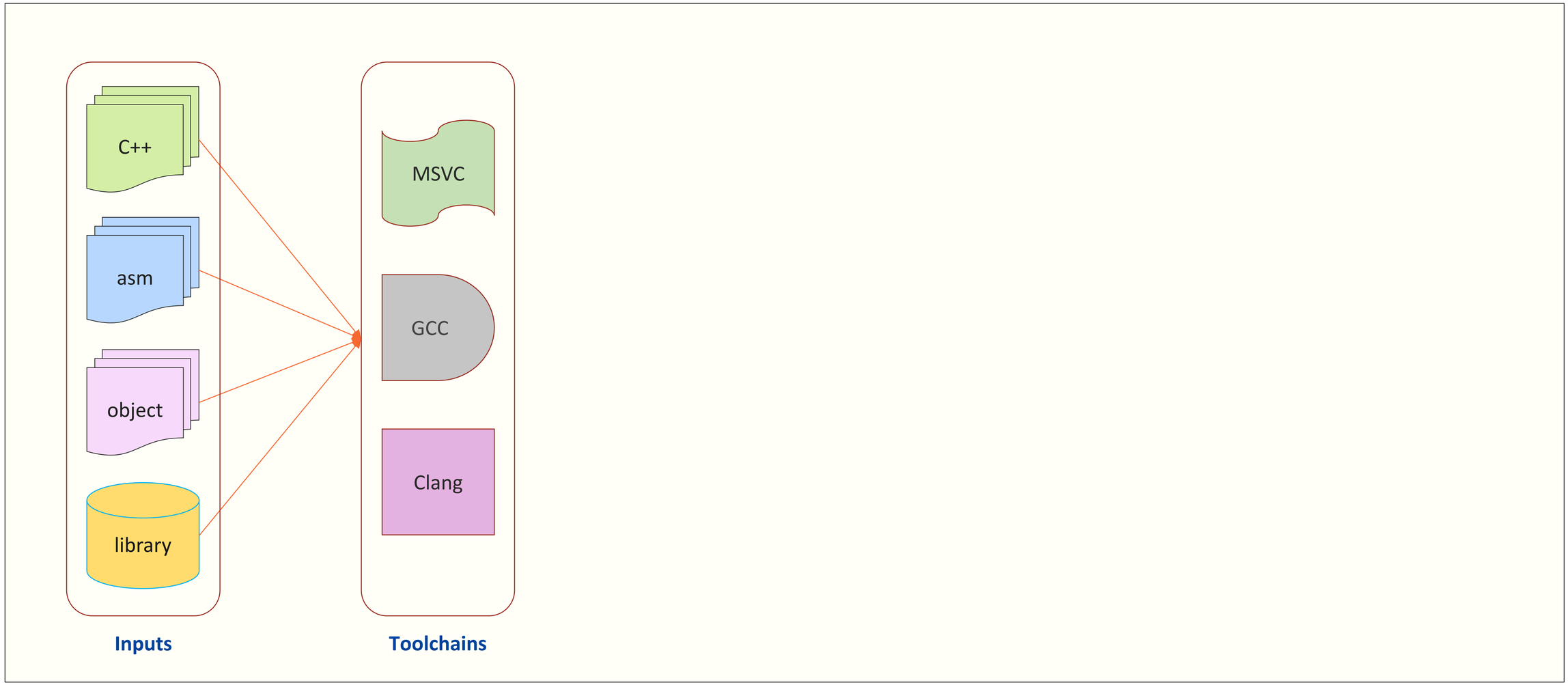
Development workflow



Inputs

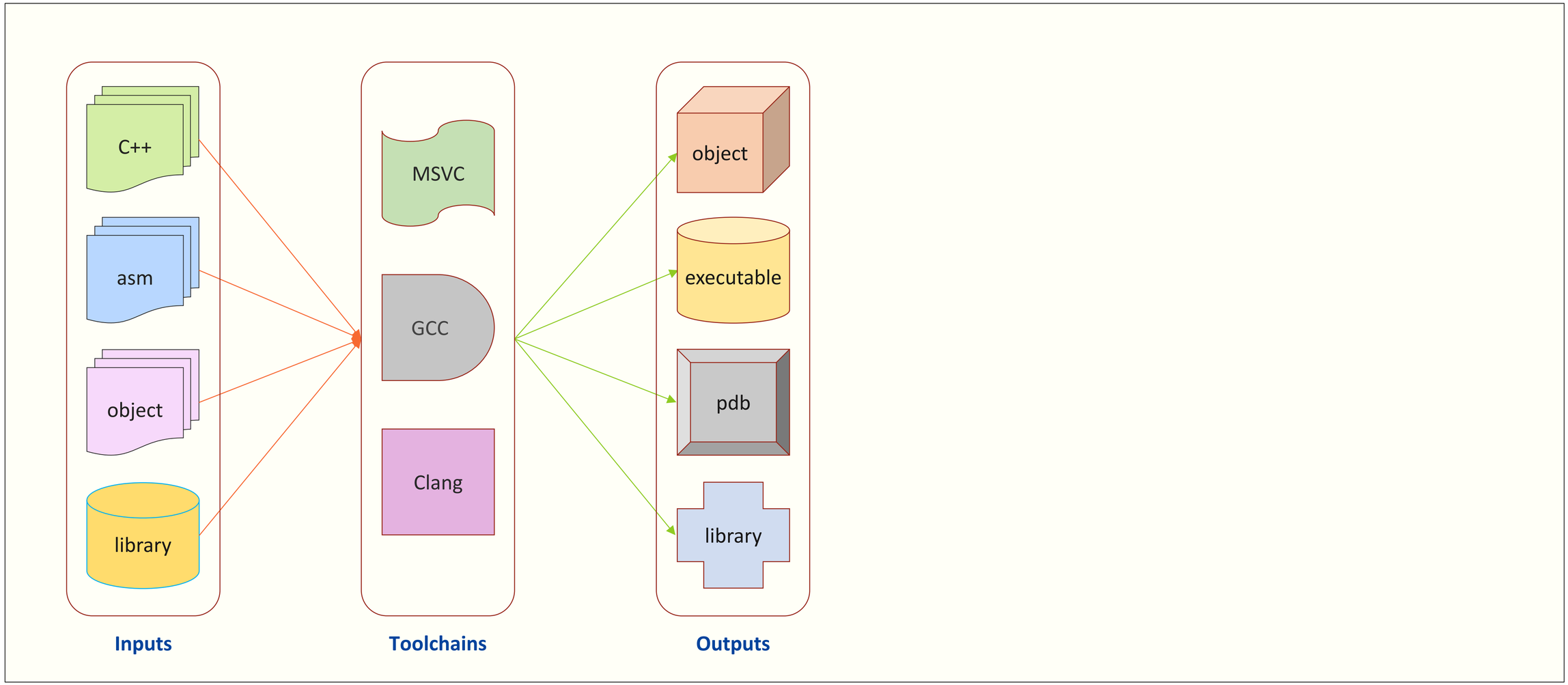
Development workflow

Development workflow



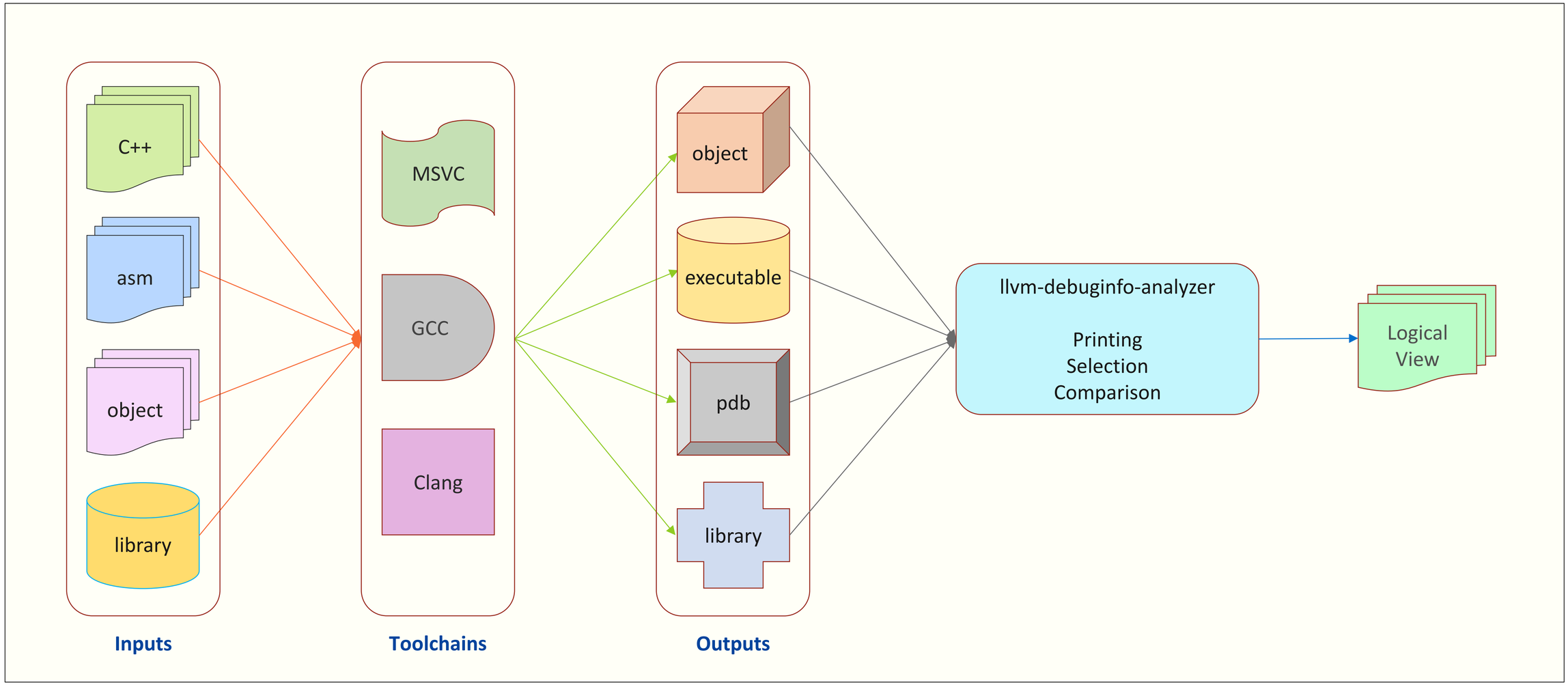
Development workflow

Development workflow



Development workflow

Development workflow



Development workflow

Logical view

```
 1 int main() {
 2   typedef unsigned TYPE;
 3   int Result = 0;
 4   {
 5     typedef float TYPE;
 6     TYPE Var_1 = 123.45;
 7     Result += Var_1;
 8   }
 9   TYPE Var_2 = 123;
10   return Result + Var_2;
11 }
```

Example use case

```
Type redefinition

Lexical Scope Level 3:
TYPE -> unsigned, declared at line 2
TYPE -> float, declared at line 5
```

```
Logical View:
[000]           {File} ‘test.out'
[001]             {CompileUnit} ‘test.cpp'
[002]     1         {Function} extern 'main' -> 'int'
[003]                 {Block}
[004]     6             {Variable} 'Var_1' -> 'TYPE'
[004]     6             {Line}
[004]     7             {Line}
[004]     7             {Line}
[003]     2           {TypeAlias} 'TYPE' -> 'unsigned'
[003]     3           {Variable} 'Result' -> 'int'
[003]     5           {TypeAlias} 'TYPE' -> 'float'
[003]     9           {Variable} 'Var_2' -> 'TYPE'
[003]     1           {Line}
[003]     3           {Line}
[003]     9           {Line}
[003]    10           {Line}
[003]    10           {Line}
[002]    10         {Line}

The logical scope at level 3, contains 2 different definitions for the
typedef 'TYPE’ with underlying type: 'unsigned’ and 'float'.
```

Logical view

Logical view

```
 1 int main() {
 2   typedef unsigned TYPE;
 3   int Result = 0;
 4   {
 5     typedef float TYPE;
 6     TYPE Var_1 = 123.45;
 7     Result += Var_1;
 8   }
 9   TYPE Var_2 = 123;
10   return Result + Var_2;
11 }
```

```
Type redefinition

Lexical Scope Level 3:
TYPE -> unsigned, declared at line 2
TYPE -> float, declared at line 5
```

```
Logical View:
[000]            {File} ‘test.out'
[001]              {CompileUnit} ‘test.cpp'
[002]     1          {Function} extern 'main' -> 'int'
[003]                  {Block}
[004]     6              {Variable} 'Var_1' -> 'TYPE'
[004]     6              {Line}
[004]     7              {Line}
[004]     7              {Line}
[003]     2            {TypeAlias} 'TYPE' -> 'unsigned'
[003]     3            {Variable} 'Result' -> 'int'
[003]     5            {TypeAlias} 'TYPE' -> 'float'
[003]     9            {Variable} 'Var_2' -> 'TYPE'
[003]     1            {Line}
[003]     3            {Line}
[003]     9            {Line}
[003]    10            {Line}
[003]    10            {Line}
[002]    10          {Line}

The logical scope at level 3, contains 2 different definitions for the
typedef 'TYPE’ with underlying type: 'unsigned’ and 'float'.
```

Example use case **Logical view**

DWARF vs Logical view vs CodeView

```
0x0000000b: DW_TAG_compile_unit
              DW_AT_name ("test.cpp")
              DW_AT_stmt_list   (0x00000000)
              DW_AT_comp_dir ("examples")
              DW_AT_low_pc   (0x00000000004004a0)
              DW_AT_high_pc  (0x00000000004004e3)
0x0000002a:   DW_TAG_subprogram
                DW_AT_low_pc (0x00000000004004a0)
                DW_AT_high_pc   (0x00000000004004e3)
                DW_AT_frame_base    (DW_OP_reg6 RBP)
                DW_AT_name   ("main")
                DW_AT_decl_file ("test.cpp")
                DW_AT_decl_line (1)
                DW_AT_type   (0x00000092 "int")
0x0000005f:     DW_TAG_lexical_block
                  DW_AT_low_pc  (0x00000000004004ba)
                  DW_AT_high_pc (0x00000000004004d4)
0x0000006c:       DW_TAG_variable
                    DW_AT_location  (DW_OP_fbreg -12)
                    DW_AT_name  ("Var_1")
                    DW_AT_decl_file ("test.cpp")
                    DW_AT_decl_line (6)
                    DW_AT_type  (0x0000007b "TYPE")
0x0000007a:       NULL
0x0000007b:     DW_TAG_typedef
                  DW_AT_type (0x00000099 "float")
                  DW_AT_name ("TYPE")
                  DW_AT_decl_file   ("test.cpp")
                  DW_AT_decl_line   (5)
```

DWARF debug information dump

```
Logical View:
[000]           {File} 'test.o'
[001]             {CompileUnit} 'test.cpp'
[002]     1         {Function} extern 'main' -> 'int'
[003]                 {Block}
[004]     6             {Variable} 'Var_1' -> 'TYPE'
[004]     6             {Line}
[004]     7             {Line}
[004]     7             {Line}
[003]     2           {TypeAlias} 'TYPE' -> 'unsigned'
[003]     3           {Variable} 'Result' -> 'int'
[003]     5           {TypeAlias} 'TYPE' -> 'float'
[003]     9           {Variable} 'Var_2' -> 'TYPE'
[003]     1           {Line}
[003]     3           {Line}
[003]     9           {Line}
[003]    10           {Line}
[003]    10           {Line}
[002]    10         {Line}
```

Logical View dump

```
Mod 0000 | `.debug$S`:
  0 | S_OBJNAME [size = 12] sig=0, ``
  0 | S_COMPILE3 [size = 48]
      machine = intel x86-x64, language = c++
  0 | S_GPROC32_ID [size = 44] `main`
      addr = 0000:0000, code size = 77
      type = `0x1002 (main)`, flags = none
  0 | S_FRAMEPROC [size = 32]
      size = 16, padding size = 0
      local fp reg = RSP, param fp reg = RSP
  0 | S_LOCAL [size = 20] `Result`
      type=0x0074 (int), flags = none
  0 | S_DEFRANGE_FRAMEPOINTER_REL [size = 16]
      offset = 8, range = [0000:0012,+65)
  0 | S_LOCAL [size = 16] `Var_2`
      type=0x0075 (unsigned), flags = none
  0 | S_DEFRANGE_FRAMEPOINTER_REL [size = 16]
      offset = 0, range = [0000:0012,+65)
  0 | S_BLOCK32 [size = 24] ``
      code size = 38, addr = 0000:0020
  0 | S_LOCAL [size = 16] `Var_1`
      type=0x0040 (float), flags = none
  0 | S_DEFRANGE_FRAMEPOINTER_REL [size = 16]
      offset = 4, range = [0000:0020,+38)
  0 | S_END [size = 4]
  0 | S_UDT [size = 20] `main::TYPE`
      original type = 0x0075 (unsigned)
  0 | S_UDT [size = 20] `main::TYPE`
      original type = 0x0040 (float)
  0 | S_PROC_ID_END [size = 4]
```

CodeView debug information dump

DWARF vs Logical view vs CodeView

```
0x0000000b: DW_TAG_compile_unit
              DW_AT_name (“test.cpp")
              DW_AT_stmt_list   (0x00000000)
              DW_AT_comp_dir ("examples")
              DW_AT_low_pc   (0x00000000004004a0)
              DW_AT_high_pc  (0x00000000004004e3)
0x0000002a:   DW_TAG_subprogram
                DW_AT_low_pc (0x00000000004004a0)
                DW_AT_high_pc   (0x00000000004004e3)
                DW_AT_frame_base    (DW_OP_reg6 RBP)
                DW_AT_name   ("main")
                DW_AT_decl_file (“test.cpp")
                DW_AT_decl_line (1)
                DW_AT_type   (0x00000092 "int")
0x0000005f:     DW_TAG_lexical_block
                  DW_AT_low_pc  (0x00000000004004ba)
                  DW_AT_high_pc (0x00000000004004d4)
0x0000006c:       DW_TAG_variable
                    DW_AT_location  (DW_OP_fbreg -12)
                    DW_AT_name  ("Var_1")
                    DW_AT_decl_file (“test.cpp")
                    DW_AT_decl_line (6)
                    DW_AT_type  (0x0000007b "TYPE")
0x0000007a:       NULL
0x0000007b:     DW_TAG_typedef
                  DW_AT_type (0x00000099 "float")
                  DW_AT_name ("TYPE")
                  DW_AT_decl_file   (“test.cpp")
                  DW_AT_decl_line   (5)
```

DWARF debug information dump

```
Logical View:
[000]           {File} ‘test.o'
[001]             {CompileUnit} ‘test.cpp'
[002]     1         {Function} extern 'main' -> 'int'
[003]               {Block}
[004]     6           {Variable} 'Var_1' -> 'TYPE'
[004]     6           {Line}
[004]     7           {Line}
[004]     7           {Line}
[003]     2         {TypeAlias} 'TYPE' -> 'unsigned'
[003]     3         {Variable} 'Result' -> 'int'
[003]     5         {TypeAlias} 'TYPE' -> 'float'
[003]     9         {Variable} 'Var_2' -> 'TYPE'
[003]     1         {Line}
[003]     3         {Line}
[003]     9         {Line}
[003]    10         {Line}
[003]    10         {Line}
[002]    10       {Line}
```

Logical View dump

```
Mod 0000 | `.debug$S`:
  0 | S_OBJNAME [size = 12] sig=0, ``
  0 | S_COMPILE3 [size = 48]
      machine = intel x86-x64, language = c++
  0 | S_GPROC32_ID [size = 44] `main`
      addr = 0000:0000, code size = 77
      type = `0x1002 (main)`, flags = none
  0 | S_FRAMEPROC [size = 32]
      size = 16, padding size = 0
      local fp reg = RSP, param fp reg = RSP
  0 | S_LOCAL [size = 20] `Result`
      type=0x0074 (int), flags = none
  0 | S_DEFRANGE_FRAMEPOINTER_REL [size = 16]
      offset = 8, range = [0000:0012,+65)
  0 | S_LOCAL [size = 16] `Var_2`
      type=0x0075 (unsigned), flags = none
  0 | S_DEFRANGE_FRAMEPOINTER_REL [size = 16]
      offset = 0, range = [0000:0012,+65)
  0 | S_BLOCK32 [size = 24] ``
      code size = 38, addr = 0000:0020
  0 | S_LOCAL [size = 16] `Var_1`
      type=0x0040 (float), flags = none
  0 | S_DEFRANGE_FRAMEPOINTER_REL [size = 16]
      offset = 4, range = [0000:0020,+38)
  0 | S_END [size = 4]
  0 | S_UDT [size = 20] `main::TYPE`
      original type = 0x0075 (unsigned)
  0 | S_UDT [size = 20] `main::TYPE`
      original type = 0x0040 (float)
  0 | S_PROC_ID_END [size = 4]
```

CodeView debug information dump

B. llvm-debuginfo-analyzer

2. Print options

Print the logical views for:

e.g.

- DWARF O0 and DWARF O2
- CodeView O0 and CodeView O2
- DWARF O0 and CodeView O0
- DWARF O2 and CodeView O2

--print (DWARF O0 and DWARF O2)

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

```
Linux: clang test.cpp –c –g –O0 –o dwarf-zero.o
Linux: clang test.cpp –c –g –O2 –o dwarf-two.o
```

Print the logical views:

```
llvm-debuginfo-analyzer
  --attribute=format
  --print=scopes,symbols,types,lines,instructions
  dwarf-zero.o dwarf-two.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=symbols,types,lines,instructions
  dwarf-zero.o dwarf-two.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=elements
  dwarf-zero.o dwarf-two.o
```

Print: DWARF O0

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
            {Code} 'movl  $0x0, -0x4(%rbp)'
 4          {Line}
            {Code} 'cmpl  $0x2, -0x4(%rbp)'
 4          {Line}
            {Code} 'jge   0x16'
 5          {Line}
            {Code} 'movl  -0x4(%rbp), %edi'
 5          {Line}
            {Code} 'callq 0x0'
 4          {Line}
            {Code} 'movl  -0x4(%rbp), %eax'
            {Code} 'addl  $0x1, %eax'
            {Code} 'movl  %eax, -0x4(%rbp)'
 4          {Line}
            {Code} 'jmp   -0x20'
 7          {Line}
            {Code} 'addq  $0x10, %rsp'
 3        {Line}
          {Code} 'pushq   %rbp'
          {Code} 'movq    %rsp, %rbp'
          {Code} 'subq    $0x10, %rsp'
          {Code} 'popq    %rbp'
          {Code} 'retq'
 7        {Line}
```

Print: DWARF O2

```
Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
            {Code} 'xorl  %edi, %edi'
            {Code} 'callq 0x0'
            {Code} 'movl  $0x1, %edi'
 5          {Line}
            {Code} 'popq  %rax'
            {Code} 'jmp   0x0'
 5          {Line}
 3        {Line}
          {Code} 'pushq   %rax'
```

--print (DWARF O0 and DWARF O2)

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

```
Linux: clang test.cpp -c -g -O0 -o dwarf-zero.o
Linux: clang test.cpp -c -g -O2 -o dwarf-two.o
```

Print the logical views:

```
llvm-debuginfo-analyzer
  --attribute=format
  --print=scopes,symbols,types,lines,instructions
  dwarf-zero.o dwarf-two.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=symbols,types,lines,instructions
  dwarf-zero.o dwarf-two.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=elements
  dwarf-zero.o dwarf-two.o
```

Print: DWARF O0

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
            {Code} 'movl  $0x0, -0x4(%rbp)'
 4          {Line}
            {Code} 'cmpl  $0x2, -0x4(%rbp)'
 4          {Line}
            {Code} 'jge   0x16'
 5          {Line}
            {Code} 'movl  -0x4(%rbp), %edi'
 5          {Line}
            {Code} 'callq 0x0'
 4          {Line}
            {Code} 'movl  -0x4(%rbp), %eax'
            {Code} 'addl  $0x1, %eax'
            {Code} 'movl  %eax, -0x4(%rbp)'
 4          {Line}
            {Code} 'jmp   -0x20'
 7          {Line}
            {Code} 'addq  $0x10, %rsp'
 3        {Line}
          {Code} 'pushq   %rbp'
          {Code} 'movq    %rsp, %rbp'
          {Code} 'subq    $0x10, %rsp'
          {Code} 'popq    %rbp'
          {Code} 'retq'
 7        {Line}
```

Print: DWARF O2

```
Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
            {Code} 'xorl  %edi, %edi'
            {Code} 'callq 0x0'
            {Code} 'movl  $0x1, %edi'
 5          {Line}
            {Code} 'popq  %rax'
            {Code} 'jmp   0x0'
 5          {Line}
 3        {Line}
          {Code} 'pushq   %rax'
```

--print (CodeView O0 and CodeView O2)

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

```
Windows: clang test.cpp –c –g –O0 –o codeview-zero.o
Windows: clang test.cpp –c –g –O2 –o codeview-two.o
```

Print the logical views:

```
llvm-debuginfo-analyzer
  --attribute=format
  --print=scopes,symbols,types,lines,instructions
  codeview-zero.o codeview-two.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=symbols,types,lines,instructions
  codeview-zero.o codeview-two.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=elements
  codeview-zero.o codeview-two.o
```

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp’
        {Function} extern not_inlined 'foo' -> 'void’
          {Block}
            {Variable} 'Index' -> 'int’
 4          {Line}
            {Code} 'movl  $0x0, 0x24(%rsp)’
            {Code} 'cmpl  $0x2, 0x24(%rsp)’
            {Code} 'jge   0x19’
 5          {Line}
            {Code} 'movl  0x24(%rsp), %ecx’
            {Code} 'callq 0x0’
 4          {Line}
            {Code} 'movl  0x24(%rsp), %eax’
            {Code} 'addl  $0x1, %eax’
            {Code} 'movl  %eax, 0x24(%rsp)’
            {Code} 'jmp   -0x24’
 3        {Line}
          {Code} 'subq    $0x28, %rsp’
 7        {Line}
          {Code} 'addq    $0x28, %rsp’
          {Code} 'retq’
```

Print: CodeView O0

```
Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp’
        {Function} extern not_inlined 'foo' -> 'void’
          {Block}
 5          {Line}
            {Code} 'xorl  %ecx, %ecx’
            {Code} 'callq 0x0’
            {Code} 'movl  $0x1, %ecx’
            {Code} 'addq  $0x28, %rsp’
            {Code} 'jmp   0x0’
 3        {Line}
          {Code} 'subq    $0x28, %rsp'
```

Print: CodeView O2

--print (CodeView O0 and CodeView O2)

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

```
Windows: clang test.cpp –c –g –O0 –o codeview-zero.o
Windows: clang test.cpp –c –g –O2 –o codeview-two.o
```

Print the logical views:

```
llvm-debuginfo-analyzer
  --attribute=format
  --print=scopes,symbols,types,lines,instructions
  codeview-zero.o codeview-two.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=symbols,types,lines,instructions
  codeview-zero.o codeview-two.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=elements
  codeview-zero.o codeview-two.o
```

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
            {Code} 'movl  $0x0, 0x24(%rsp)'
            {Code} 'cmpl  $0x2, 0x24(%rsp)'
            {Code} 'jge   0x19'
 5          {Line}
            {Code} 'movl  0x24(%rsp), %ecx'
            {Code} 'callq 0x0'
 4          {Line}
            {Code} 'movl  0x24(%rsp), %eax'
            {Code} 'addl  $0x1, %eax'
            {Code} 'movl  %eax, 0x24(%rsp)'
            {Code} 'jmp   -0x24'
 3        {Line}
          {Code} 'subq    $0x28, %rsp'
 7        {Line}
          {Code} 'addq    $0x28, %rsp'
          {Code} 'retq'
```

Print: CodeView O0

```
Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
            {Code} 'xorl  %ecx, %ecx'
            {Code} 'callq 0x0'
            {Code} 'movl  $0x1, %ecx'
            {Code} 'addq  $0x28, %rsp'
            {Code} 'jmp   0x0'
 3        {Line}
          {Code} 'subq    $0x28, %rsp'
```

Print: CodeView O2

--print (DWARF O0 and CodeView O0)

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Print the logical views:

```
llvm-debuginfo-analyzer
  --attribute=format
  --print=scopes,symbols,types,lines,instructions
  dwarf-zero.o codeview-zero.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=symbols,types,lines,instructions
  dwarf-zero.o codeview-zero.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=elements
  dwarf-zero.o codeview-zero.o
```

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
            {Code} 'movl  $0x0, -0x4(%rbp)'
 4          {Line}
            {Code} 'cmpl  $0x2, -0x4(%rbp)'
 4          {Line}
            {Code} 'jge   0x16'
 5          {Line}
            {Code} 'movl  -0x4(%rbp), %edi'
 5          {Line}
            {Code} 'callq 0x0'
 4          {Line}
            {Code} 'movl  -0x4(%rbp), %eax'
            {Code} 'addl  $0x1, %eax'
            {Code} 'movl  %eax, -0x4(%rbp)'
 4          {Line}
            {Code} 'jmp   -0x20'
 7          {Line}
            {Code} 'addq  $0x10, %rsp'
 3        {Line}
          {Code} 'pushq   %rbp'
          {Code} 'movq    %rsp, %rbp'
          {Code} 'subq    $0x10, %rsp'
          {Code} 'popq    %rbp'
          {Code} 'retq'
 7        {Line}
```

Print: DWARF O0

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
            {Code} 'movl  $0x0, 0x24(%rsp)'
            {Code} 'cmpl  $0x2, 0x24(%rsp)'
            {Code} 'jge   0x19'
 5          {Line}
            {Code} 'movl  0x24(%rsp), %ecx'
            {Code} 'callq 0x0'
 4          {Line}
            {Code} 'movl  0x24(%rsp), %eax'
            {Code} 'addl  $0x1, %eax'
            {Code} 'movl  %eax, 0x24(%rsp)'
            {Code} 'jmp   -0x24'
 3        {Line}
          {Code} 'subq    $0x28, %rsp'
 7        {Line}
          {Code} 'addq    $0x28, %rsp'
          {Code} 'retq'
```

Print: CodeView O0

--print (DWARF O0 and CodeView O0)

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Print the logical views:

```
llvm-debuginfo-analyzer
  --attribute=format
  --print=scopes,symbols,types,lines,instructions
  dwarf-zero.o codeview-zero.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=symbols,types,lines,instructions
  dwarf-zero.o codeview-zero.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=elements
  dwarf-zero.o codeview-zero.o
```

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
3       {Function} extern not_inlined 'foo' -> 'void'
          {Block}
4           {Variable} 'Index' -> 'int'
4           {Line}
            {Code} 'movl  $0x0, -0x4(%rbp)'
4           {Line}
            {Code} 'cmpl  $0x2, -0x4(%rbp)'
4           {Line}
            {Code} 'jge   0x16'
5           {Line}
            {Code} 'movl  -0x4(%rbp), %edi'
5           {Line}
            {Code} 'callq 0x0'
4           {Line}
            {Code} 'movl  -0x4(%rbp), %eax'
            {Code} 'addl  $0x1, %eax'
            {Code} 'movl  %eax, -0x4(%rbp)'
4           {Line}
            {Code} 'jmp   -0x20'
7           {Line}
            {Code} 'addq  $0x10, %rsp'
3         {Line}
          {Code} 'pushq   %rbp'
          {Code} 'movq    %rsp, %rbp'
          {Code} 'subq    $0x10, %rsp'
          {Code} 'popq    %rbp'
          {Code} 'retq'
7         {Line}
```

Print: DWARF O0

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
4           {Line}
            {Code} 'movl  $0x0, 0x24(%rsp)'
            {Code} 'cmpl  $0x2, 0x24(%rsp)'
            {Code} 'jge   0x19'
5           {Line}
            {Code} 'movl  0x24(%rsp), %ecx'
            {Code} 'callq 0x0'
4           {Line}
            {Code} 'movl  0x24(%rsp), %eax'
            {Code} 'addl  $0x1, %eax'
            {Code} 'movl  %eax, 0x24(%rsp)'
            {Code} 'jmp   -0x24'
3         {Line}
          {Code} 'subq    $0x28, %rsp'
7         {Line}
          {Code} 'addq    $0x28, %rsp'
          {Code} 'retq'
```

Print: CodeView O0

--print (DWARF O2 and CodeView O2)

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Print the logical views:

```
llvm-debuginfo-analyzer
  --attribute=format
  --print=scopes,symbols,types,lines,instructions
  dwarf-two.o codeview-two.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=symbols,types,lines,instructions
  dwarf-two.o codeview-two.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=elements
  dwarf-two.o codeview-two.o
```

Print: DWARF O2

```
Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
            {Code} 'xorl  %edi, %edi'
            {Code} 'callq 0x0'
            {Code} 'movl  $0x1, %edi'
 5          {Line}
            {Code} 'popq  %rax'
            {Code} 'jmp   0x0'
 5          {Line}
 3        {Line}
          {Code} 'pushq   %rax'
```

Print: CodeView O2

```
Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
            {Code} 'xorl  %ecx, %ecx'
            {Code} 'callq 0x0'
            {Code} 'movl  $0x1, %ecx'
            {Code} 'addq  $0x28, %rsp'
            {Code} 'jmp   0x0'
 3        {Line}
          {Code} 'subq    $0x28, %rsp'
```

--print (DWARF O2 and CodeView O2)

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Print the logical views:

```
llvm-debuginfo-analyzer
  --attribute=format
  --print=scopes,symbols,types,lines,instructions
  dwarf-two.o codeview-two.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=symbols,types,lines,instructions
  dwarf-two.o codeview-two.o

or

llvm-debuginfo-analyzer
  --attribute=format
  --print=elements
  dwarf-two.o codeview-two.o
```

Print: DWARF O2

```
Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
            {Code} 'xorl  %edi, %edi'
            {Code} 'callq 0x0'
            {Code} 'movl  $0x1, %edi'
 5          {Line}
            {Code} 'popq  %rax'
            {Code} 'jmp   0x0'
 5          {Line}
 3        {Line}
          {Code} 'pushq   %rax'
```

Print: CodeView O2

```
Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
            {Code} 'xorl  %ecx, %ecx'
            {Code} 'callq 0x0'
            {Code} 'movl  $0x1, %ecx'
            {Code} 'addq  $0x28, %rsp'
            {Code} 'jmp   0x0'
 3        {Line}
          {Code} 'subq    $0x28, %rsp'
```

B. llvm-debuginfo-analyzer

3. Select options

Print selected logical elements using **single criteria** (list and view layout) for:

e.g.

- DWARF O0 and DWARF O2
- CodeView O0 and CodeView O2

Print selected logical elements using **combined criteria** (list and view layout) for:

e.g.

- DWARF O0 and DWARF O2
- CodeView O0 and CodeView O2

--select (DWARF O0 and DWARF O2) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Select logical elements:

Selection criteria:
  Index or 4

```
llvm-debuginfo-analyzer
  --select=Index --select=4
  --report=list
  --print=symbols,lines
  dwarf-zero.o dwarf-two.o
```

DWARF O0 and DWARF O2

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}
 4          {Line}
 5          {Line}
 5          {Line}
 4          {Line}
 4          {Line}
 7          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}
```

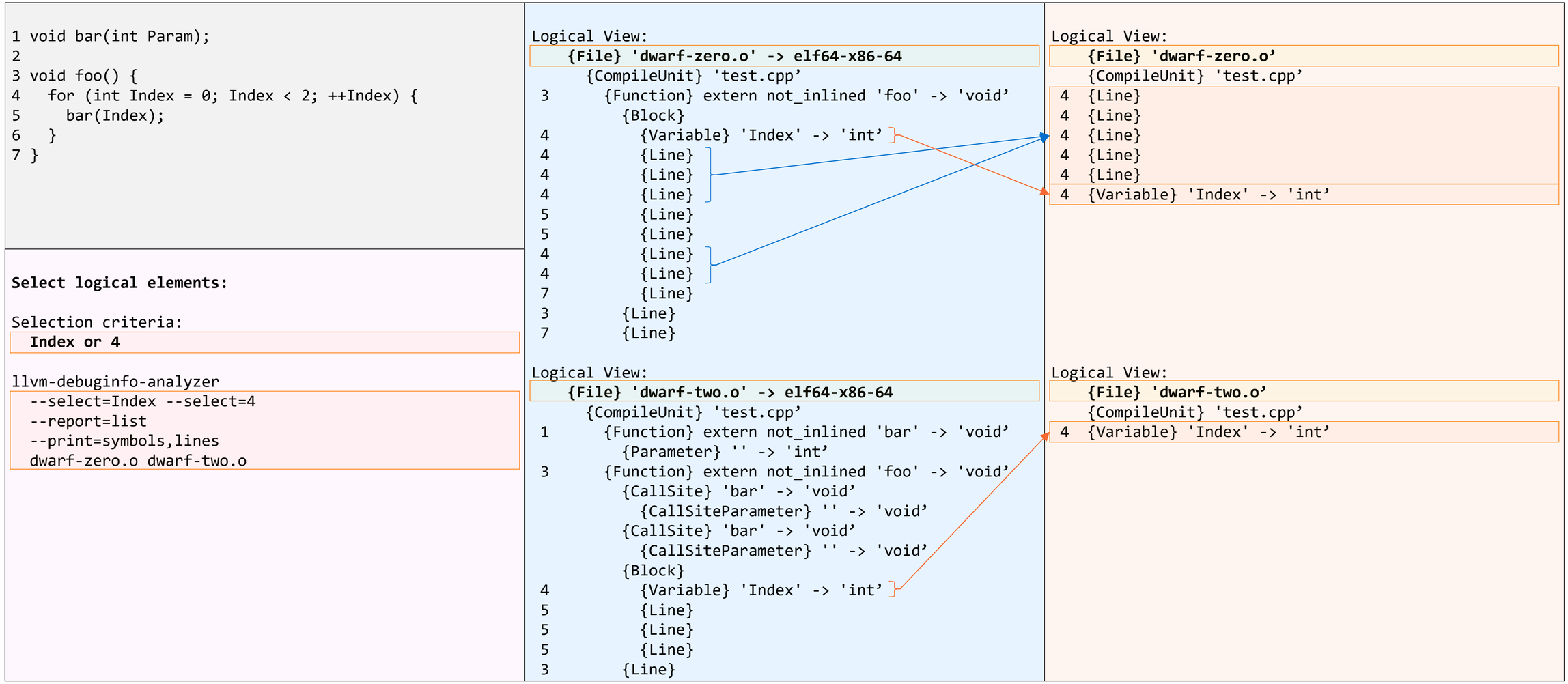
Print

```
Logical View:
    {File} 'dwarf-zero.o'
    {CompileUnit} 'test.cpp'
 4  {Line}
 4  {Line}
 4  {Line}
 4  {Line}
 4  {Line}
 4  {Variable} 'Index' -> 'int'

Logical View:
    {File} 'dwarf-two.o'
    {CompileUnit} 'test.cpp'
 4  {Variable} 'Index' -> 'int'
```

Select: list layout

--select (DWARF O0 and DWARF O2) - list layout



--select (DWARF O0 and DWARF O2) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Select logical elements:

Selection criteria:
  Index or 4

```
llvm-debuginfo-analyzer
  --select=Index --select=4
  --report=view
  --print=symbols,lines
  dwarf-zero.o dwarf-two.o
```

DWARF O0 and DWARF O2

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}
 4          {Line}
 5          {Line}
 5          {Line}
 4          {Line}
 4          {Line}
 7          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}
```

Print

```
Logical View:
    {File} 'dwarf-zero.o'
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}
 4          {Line}
 4          {Line}
 4          {Line}
 4          {Line}

Logical View:
    {File} 'dwarf-two.o'
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
```

Select: view layout

--select (DWARF O0 and DWARF O2) - view layout

DWARF O0 and DWARF O2 | Print | Select: view layout

--select (CodeView O0 and CodeView O2) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Select logical elements:

Selection criteria:
  Index or 4

```
llvm-debuginfo-analyzer
  --select=Index --select=4
  --report=list
  --print=symbols,lines
  codeview-zero.o codeview-two.o
```

CodeView O0 and CodeView O2

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Print

```
Logical View:
    {File} 'codeview-zero.o'
    {CompileUnit} 'test.cpp'
    {Variable} 'Index' -> 'int'
 4  {Line}
 4  {Line}

Logical View:
    {File} 'codeview-two.o'
    {CompileUnit} 'test.cpp'
```

Select: list layout

--select (CodeView O0 and CodeView O2) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Select logical elements:

Selection criteria:

```
  Index or 4
```

```
llvm-debuginfo-analyzer
  --select=Index --select=4
  --report=list
  --print=symbols,lines
  codeview-zero.o codeview-two.o
```

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

```
Logical View:
    {File} 'codeview-zero.o'
      {CompileUnit} 'test.cpp'
      {Variable} 'Index' -> 'int'
 4    {Line}
 4    {Line}

Logical View:
    {File} 'codeview-two.o'
      {CompileUnit} 'test.cpp'
```

CodeView O0 and CodeView O2 | **Print** | **Select: list layout**

--select (CodeView O0 and CodeView O2) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Select logical elements:

Selection criteria:
  Index or 4

```
llvm-debuginfo-analyzer
  --select=Index --select=4
  --report=view
  --print=symbols,lines
  codeview-zero.o codeview-two.o
```

CodeView O0 and CodeView O2

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Print

```
Logical View:
    {File} 'codeview-zero.o'
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}

Logical View:
    {File} 'codeview-two.o'
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
```

Select: view layout

--select (CodeView O0 and CodeView O2) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Select logical elements:

Selection criteria:

```
  Index or 4
```

```
llvm-debuginfo-analyzer
  --select=Index --select=4
  --report=view
  --print=symbols,lines
  codeview-zero.o codeview-two.o
```

CodeView O0 and CodeView O2

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Print

```
Logical View:
    {File} 'codeview-zero.o'
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}

Logical View:
    {File} 'codeview-two.o'
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
```

Select: view layout

--select (combined) (DWARF O0 and DWARF O2) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Select logical elements:

```
Combined selection criteria:
  LineDebug and 4

llvm-debuginfo-analyzer
  --select-lines=LineDebug --select=4
  --report=list
  --print=symbols,lines
  dwarf-zero.o

Combined selection criteria:
  Variable and Index

llvm-debuginfo-analyzer
  --select-symbols=Variable --select=Index
  --report=list
  --print=symbols,lines
  dwarf-two.o
```

DWARF O0 and DWARF O2

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}
 4          {Line}
 5          {Line}
 5          {Line}
 4          {Line}
 4          {Line}
 7          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}
```

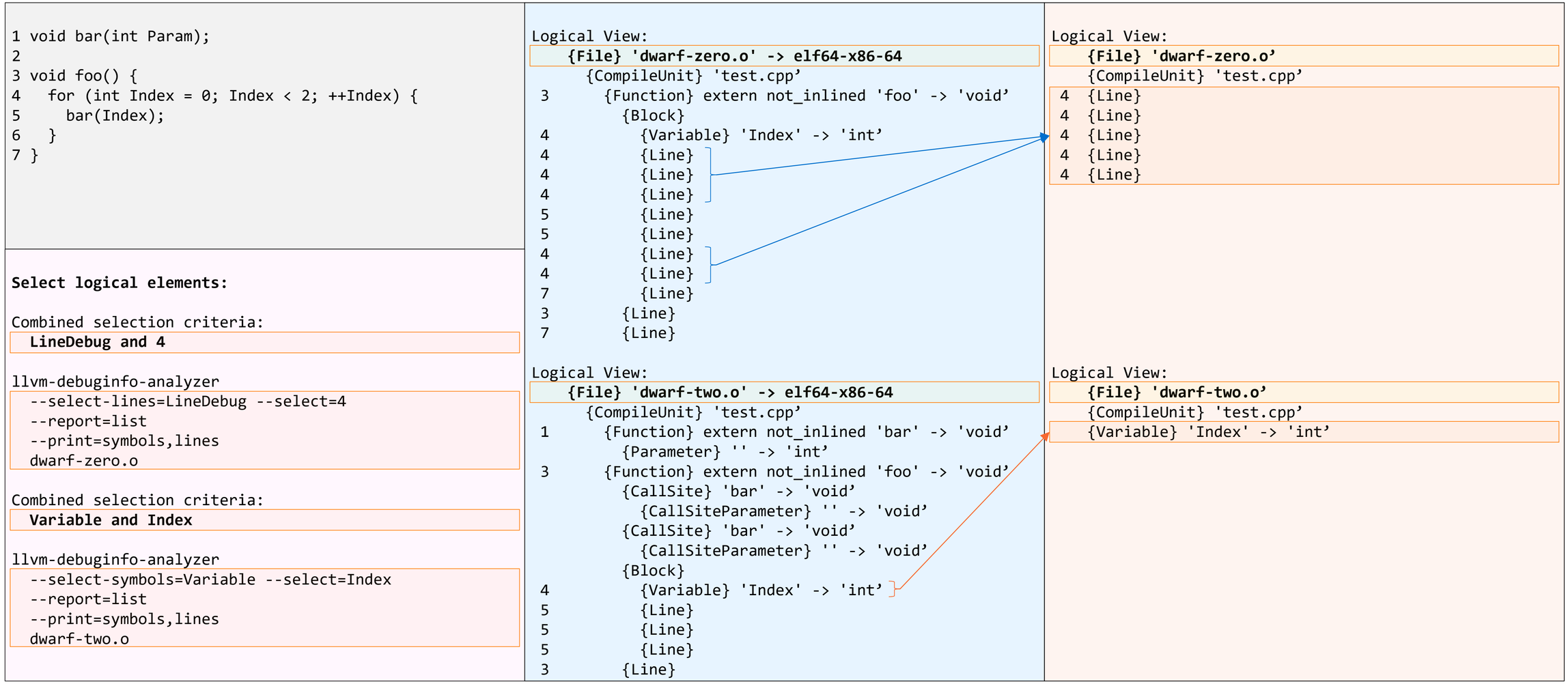
Print

```
Logical View:
    {File} 'dwarf-zero.o'
    {CompileUnit} 'test.cpp'
 4  {Line}
 4  {Line}
 4  {Line}
 4  {Line}
 4  {Line}

Logical View:
    {File} 'dwarf-two.o'
    {CompileUnit} 'test.cpp'
    {Variable} 'Index' -> 'int'
```

Combined select: list layout

--select (combined) (DWARF O0 and DWARF O2) - list layout



DWARF O0 and DWARF O2 | **Print** | **Combined select: list layout**

--select (combined) (DWARF O0 and DWARF O2) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Select logical elements:

```
Combined selection criteria:
  LineDebug and 4

llvm-debuginfo-analyzer
  --select-lines=LineDebug --select=4
  --report=view
  --print=symbols,lines
  dwarf-zero.o

Combined selection criteria:
  Variable and Index

llvm-debuginfo-analyzer
  --select-symbols=Variable --select=Index
  --report=view
  --print=symbols,lines
  dwarf-two.o
```

DWARF O0 and DWARF O2

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}
 4          {Line}
 5          {Line}
 5          {Line}
 4          {Line}
 4          {Line}
 7          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}
```

Print

```
Logical View:
    {File} 'dwarf-zero.o'
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Line}
 4          {Line}
 4          {Line}
 4          {Line}
 4          {Line}

Logical View:
    {File} 'dwarf-two.o'
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
```

Combined select: view layout

--select (combined) (DWARF O0 and DWARF O2) - view layout

DWARF O0 and DWARF O2 | **Print** | **Combined select: view layout**

--select (combined) (CodeView O0 and CodeView O2) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Select logical elements:

```
Combined selection criteria:
  LineDebug and 4

llvm-debuginfo-analyzer
  --select-lines=LineDebug --select=4
  --report=list
  --print=symbols,lines
  codeview-zero.o

Combined selection criteria:
  Variable and Index

llvm-debuginfo-analyzer
  --select-symbols=Variable --select=Index
  --report=list
  --print=symbols,lines
  codeview-two.o
```

CodeView O0 and CodeView O2

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Print

```
Logical View:
    {File} 'codeview-zero.o'
    {CompileUnit} 'test.cpp'
 4  {Line}
 4  {Line}

Logical View:
    {File} 'codeview-two.o'
    {CompileUnit} 'test.cpp'
```

Combined select: list layout

--select (combined) (CodeView O0 and CodeView O2) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Select logical elements:

Combined selection criteria:

```
  LineDebug and 4
```

```
llvm-debuginfo-analyzer
  --select-lines=LineDebug --select=4
  --report=list
  --print=symbols,lines
  codeview-zero.o
```

Combined selection criteria:

```
  Variable and Index
```

```
llvm-debuginfo-analyzer
  --select-symbols=Variable --select=Index
  --report=list
  --print=symbols,lines
  codeview-two.o
```

CodeView O0 and CodeView O2

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Print

```
Logical View:
    {File} 'codeview-zero.o'
    {CompileUnit} 'test.cpp'
 4  {Line}
 4  {Line}

Logical View:
    {File} 'codeview-two.o'
    {CompileUnit} 'test.cpp'
```

Combined select: list layout

--select (combined) (CodeView O0 and CodeView O2) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Select logical elements:

Combined selection criteria:
  LineDebug and 4

```
llvm-debuginfo-analyzer
  --select-lines=LineDebug --select=4
  --report=view
  --print=symbols,lines
  codeview-zero.o
```

Combined selection criteria:
  Variable and Index

```
llvm-debuginfo-analyzer
  --select-symbols=Variable --select=Index
  --report=view
  --print=symbols,lines
  codeview-two.o
```

CodeView O0 and CodeView O2

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Print

```
Logical View:
    {File} 'codeview-zero.o'
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}

Logical View:
    {File} 'codeview-two.o'
      {CompileUnit} 'test.cpp'
```

Combined select: view layout

--select (combined) (CodeView O0 and CodeView O2) - view layout

CodeView O0 and CodeView O2 | **Print** | **Combined select: view layout**

B. llvm-debuginfo-analyzer

4. Compare options

Find semantic differences by comparing logical views (list and view layout) for:

e.g.

- DWARF O0 and DWARF O2
- CodeView O0 and CodeView O2
- DWARF O0 and CodeView O0
- DWARF O2 and CodeView O2

--compare (DWARF O0 and DWARF O2) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=list
  --print=symbols,lines
  dwarf-zero.o dwarf-two.o
```

DWARF O0 and DWARF O2

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}
 4          {Line}
 5          {Line}
 5          {Line}
 4          {Line}
 4          {Line}
 7          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}
```

Print

```
Reference: 'dwarf-zero.o'
Target:    'dwarf-two.o'

(7) Missing Lines:
-   4   {Line}
-   4   {Line}
-   4   {Line}
-   4   {Line}
-   4   {Line}
-   7   {Line}
-   7   {Line}

(3) Added Scopes:
+       {CallSite} 'bar' -> 'void'
+       {CallSite} 'bar' -> 'void'
+   1   {Function} extern not_inlined 'bar' -> 'void'

(5) Added Symbols:
+       {CallSiteParameter} '' -> 'void'
+       {CallSiteParameter} '' -> 'void'
+       {Parameter} '' -> 'int'
+       {Parameter} '' -> 'int'
+       {Parameter} '' -> 'int'
```

Compare: list layout

--compare (DWARF O0 and DWARF O2) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=list
  --print=symbols,lines
  dwarf-zero.o dwarf-two.o
```

DWARF O0 and DWARF O2

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}
 4          {Line}
 5          {Line}
 5          {Line}
 4          {Line}
 4          {Line}
 7          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}
```

Print

```
Reference: 'dwarf-zero.o'
Target:    'dwarf-two.o'

(7) Missing Lines:
-   4   {Line}
-   4   {Line}
-   4   {Line}
-   4   {Line}
-   4   {Line}
-   7   {Line}
-   7   {Line}

(3) Added Scopes:
+       {CallSite} 'bar' -> 'void'
+       {CallSite} 'bar' -> 'void'
+   1   {Function} extern not_inlined 'bar' -> 'void'

(5) Added Symbols:
+       {CallSiteParameter} '' -> 'void'
+       {CallSiteParameter} '' -> 'void'
+       {Parameter} '' -> 'int'
+       {Parameter} '' -> 'int'
+       {Parameter} '' -> 'int'
```

Compare: list layout

--compare (DWARF O0 and DWARF O2) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=view
  --print=symbols,lines
  dwarf-zero.o dwarf-two.o
```

DWARF O0 and DWARF O2

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}
 4          {Line}
 5          {Line}
 5          {Line}
 4          {Line}
 4          {Line}
 7          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}
```

Print

```
Reference: 'dwarf-zero.o'
Target:    'dwarf-two.o'

Logical View:
    {File} 'dwarf-zero.o'
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
-4          {Line}
-4          {Line}
-4          {Line}
 5          {Line}
 5          {Line}
-4          {Line}
-4          {Line}
-7          {Line}
+          {CallSite} 'bar' -> 'void'
+            {CallSiteParameter} '' -> 'void'
+            {Parameter} '' -> 'int'
+          {CallSite} 'bar' -> 'void'
+            {CallSiteParameter} '' -> 'void'
+            {Parameter} '' -> 'int'
 3        {Line}
-7        {Line}
+1      {Function} extern not_inlined 'bar' -> 'void'
+          {Parameter} '' -> 'int'
```

Compare: view layout

--compare (DWARF O0 and DWARF O2) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=view
  --print=symbols,lines
  dwarf-zero.o dwarf-two.o
```

DWARF O0 and DWARF O2

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}
 4          {Line}
 5          {Line}
 5          {Line}
 4          {Line}
 4          {Line}
 7          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}
```

Print

```
Reference: 'dwarf-zero.o'
Target:    'dwarf-two.o'

Logical View:
    {File} 'dwarf-zero.o'
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
-4          {Line}
-4          {Line}
-4          {Line}
 5          {Line}
 5          {Line}
-4          {Line}
-4          {Line}
-7          {Line}
+          {CallSite} 'bar' -> 'void'
+            {CallSiteParameter} '' -> 'void'
+            {Parameter} '' -> 'int'
+          {CallSite} 'bar' -> 'void'
+            {CallSiteParameter} '' -> 'void'
+            {Parameter} '' -> 'int'
 3        {Line}
-7        {Line}
+1      {Function} extern not_inlined 'bar' -> 'void'
+          {Parameter} '' -> 'int'
```

Compare: view layout

--compare (CodeView O0 and CodeView O2) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=list
  --print=symbols,lines
  codeview-zero.o codeview-two.o
```

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

```
Reference: 'codeview-zero.o'
Target:    'codeview-two.o'

(3) Missing Lines:
-   4   {Line}
-   4   {Line}
-   7   {Line}
```

CodeView O0 and CodeView O2 | **Print** | **Compare: list layout**

--compare (CodeView O0 and CodeView O2) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=list
  --print=symbols,lines
  codeview-zero.o codeview-two.o
```

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

```
Reference: 'codeview-zero.o'
Target:    'codeview-two.o'

(3) Missing Lines:
-   4   {Line}
-   4   {Line}
-   7   {Line}
```

CodeView O0 and CodeView O2 | **Print** | **Compare: list layout**

--compare (CodeView O0 and CodeView O2) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=view
  --print=symbols,lines
  codeview-zero.o codeview-two.o
```

CodeView O0 and CodeView O2

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Print

```
Reference: 'codeview-zero.o'
Target:    'codeview-two.o'

Logical View:
    {File} 'codeview-zero.o'
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
-4          {Line}
 5          {Line}
-4          {Line}
 3        {Line}
-7        {Line}
```

Compare: view layout

--compare (CodeView O0 and CodeView O2) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=view
  --print=symbols,lines
  codeview-zero.o codeview-two.o
```

CodeView O0 and CodeView O2

```
Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Print

```
Reference: 'codeview-zero.o'
Target:    'codeview-two.o'

Logical View:
    {File} 'codeview-zero.o'
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
-4          {Line}
 5          {Line}
-4          {Line}
 3        {Line}
-7        {Line}
```

Compare: view layout

--compare (DWARF O0 and CodeView O0) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=list
  --print=symbols,lines
  dwarf-zero.o codeview-zero.o
```

DWARF O0 and CodeView O0

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
3       {Function} extern not_inlined 'foo' -> 'void'
          {Block}
4           {Variable} 'Index' -> 'int'
4           {Line}
4           {Line}
4           {Line}
5           {Line}
5           {Line}
4           {Line}
4           {Line}
7           {Line}
3         {Line}
7         {Line}

Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
4           {Line}
5           {Line}
4           {Line}
3         {Line}
7         {Line}
```

Print

```
Reference: 'dwarf-zero.o'
Target:    'codeview-zero.o'

(1) Missing Scopes:
-   3   {Function} extern not_inlined 'foo' -> 'void'

(1) Missing Symbols:
-   4   {Variable} 'Index' -> 'int'

(1) Missing Lines:
-   7   {Line}

(1) Added Scopes:
+       {Function} extern not_inlined 'foo' -> 'void'

(1) Added Symbols:
+       {Variable} 'Index' -> 'int'
```

Compare: list layout

--compare (DWARF O0 and CodeView O0) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=list
  --print=symbols,lines
  dwarf-zero.o codeview-zero.o
```

DWARF O0 and CodeView O0

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
3       {Function} extern not_inlined 'foo' -> 'void'
          {Block}
4           {Variable} 'Index' -> 'int'
4           {Line}
4           {Line}
4           {Line}
5           {Line}
5           {Line}
4           {Line}
4           {Line}
7           {Line}
3         {Line}
7         {Line}

Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
4           {Line}
5           {Line}
4           {Line}
3         {Line}
7         {Line}
```

Print

```
Reference: 'dwarf-zero.o'
Target:    'codeview-zero.o'

(1) Missing Scopes:
-   3   {Function} extern not_inlined 'foo' -> 'void'

(1) Missing Symbols:
-   4   {Variable} 'Index' -> 'int'

(1) Missing Lines:
-   7   {Line}

(1) Added Scopes:
+       {Function} extern not_inlined 'foo' -> 'void'

(1) Added Symbols:
+       {Variable} 'Index' -> 'int'
```

Compare: list layout

--compare (DWARF O0 and CodeView O0) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=view
  --print=symbols,lines
  dwarf-zero.o codeview-zero.o
```

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}
 4          {Line}
 5          {Line}
 5          {Line}
 4          {Line}
 4          {Line}
 7          {Line}
 3        {Line}
 7        {Line}

Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}
```

```
Reference: 'dwarf-zero.o'
Target:    'codeview-zero.o'

Logical View:
    {File} 'dwarf-zero.o'
      {CompileUnit} 'test.cpp'
-3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
-4          {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}
 4          {Line}
 5          {Line}
 5          {Line}
 4          {Line}
 4          {Line}
-7          {Line}
 3        {Line}
 7        {Line}
+       {Function} extern not_inlined 'foo' -> 'void'
          {Block}
+           {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}
```

DWARF O0 and CodeView O0 | **Print** | **Compare: view layout**

--compare (DWARF O0 and CodeView O0) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=view
  --print=symbols,lines
  dwarf-zero.o codeview-zero.o
```

DWARF O0 and CodeView O0

```
Logical View:
    {File} 'dwarf-zero.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
3       {Function} extern not_inlined 'foo' -> 'void'
          {Block}
4           {Variable} 'Index' -> 'int'
4           {Line}
4           {Line}
4           {Line}
5           {Line}
5           {Line}
4           {Line}
4           {Line}
7           {Line}
3         {Line}
7         {Line}

Logical View:
    {File} 'codeview-zero.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
            {Variable} 'Index' -> 'int'
4           {Line}
5           {Line}
4           {Line}
3         {Line}
7         {Line}
```

Print

```
Reference: 'dwarf-zero.o'
Target:    'codeview-zero.o'

Logical View:
    {File} 'dwarf-zero.o'
      {CompileUnit} 'test.cpp'
-3      {Function} extern not_inlined 'foo' -> 'void'
          {Block}
-4          {Variable} 'Index' -> 'int'
 4          {Line}
 4          {Line}
 4          {Line}
 5          {Line}
 5          {Line}
 4          {Line}
 4          {Line}
-7          {Line}
 3        {Line}
 7        {Line}
+       {Function} extern not_inlined 'foo' -> 'void'
          {Block}
+           {Variable} 'Index' -> 'int'
 4          {Line}
 5          {Line}
 4          {Line}
 3        {Line}
 7        {Line}
```

Compare: view layout

--compare (DWARF O2 and CodeView O2) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=list
  --print=symbols,lines
  dwarf-two.o codeview-two.o
```

DWARF O2 and CodeView O2

```
Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Print

```
Reference: 'dwarf-two.o'
Target:    'codeview-two.o'

(4) Missing Scopes:
-   3   {Function} extern not_inlined 'foo' -> 'void'
-       {CallSite} 'bar' -> 'void'
-       {CallSite} 'bar' -> 'void'
-   1   {Function} extern not_inlined 'bar' -> 'void'

(6) Missing Symbols:
-   4   {Variable} 'Index' -> 'int'
-       {CallSiteParameter} '' -> 'void'
-       {CallSiteParameter} '' -> 'void'
-       {Parameter} '' -> 'int'
-       {Parameter} '' -> 'int'
-       {Parameter} '' -> 'int'

(1) Added Scopes:
+       {Function} extern not_inlined 'foo' -> 'void'
```

Compare: list layout

--compare (DWARF O2 and CodeView O2) - list layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=list
  --print=symbols,lines
  dwarf-two.o codeview-two.o
```

DWARF O2 and CodeView O2

```
Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Print

```
Reference: 'dwarf-two.o'
Target:    'codeview-two.o'

(4) Missing Scopes:
-   3   {Function} extern not_inlined 'foo' -> 'void'
-       {CallSite} 'bar' -> 'void'
-       {CallSite} 'bar' -> 'void'
-   1   {Function} extern not_inlined 'bar' -> 'void'

(6) Missing Symbols:
-   4   {Variable} 'Index' -> 'int'
-       {CallSiteParameter} '' -> 'void'
-       {CallSiteParameter} '' -> 'void'
-       {Parameter} '' -> 'int'
-       {Parameter} '' -> 'int'
-       {Parameter} '' -> 'int'

(1) Added Scopes:
+       {Function} extern not_inlined 'foo' -> 'void'
```

Compare: list layout

--compare (DWARF O2 and CodeView O2) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=view
  --print=symbols,lines
  dwarf-two.o codeview-two.o
```

DWARF O2 and CodeView O2

```
Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Print

```
Reference: 'dwarf-two.o'
Target:    'codeview-two.o'

Logical View:
    {File} 'dwarf-two.o'
      {CompileUnit} 'test.cpp'
-1      {Function} extern not_inlined 'bar' -> 'void'
-         {Parameter} '' -> 'int'
-3      {Function} extern not_inlined 'foo' -> 'void'
-         {CallSite} 'bar' -> 'void'
-           {CallSiteParameter} '' -> 'void'
-           {Parameter} '' -> 'int'
-         {CallSite} 'bar' -> 'void'
-           {CallSiteParameter} '' -> 'void'
-           {Parameter} '' -> 'int'
-         {Block}
-4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}
+       {Function} extern not_inlined 'foo' -> 'void'
           {Block}
 5          {Line}
 3        {Line}
```

Compare: view layout

--compare (DWARF O2 and CodeView O2) - view layout

```
1 void bar(int Param);
2
3 void foo() {
4   for (int Index = 0; Index < 2; ++Index) {
5     bar(Index);
6   }
7 }
```

Compare the logical views:

```
llvm-debuginfo-analyzer
  --compare=symbols,lines
  --report=view
  --print=symbols,lines
  dwarf-two.o codeview-two.o
```

DWARF O2 and CodeView O2

```
Logical View:
    {File} 'dwarf-two.o' -> elf64-x86-64
      {CompileUnit} 'test.cpp'
 1      {Function} extern not_inlined 'bar' -> 'void'
          {Parameter} '' -> 'int'
 3      {Function} extern not_inlined 'foo' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {CallSite} 'bar' -> 'void'
            {CallSiteParameter} '' -> 'void'
          {Block}
 4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}

Logical View:
    {File} 'codeview-two.o' -> COFF-x86-64
      {CompileUnit} 'test.cpp'
        {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Print

```
Reference: 'dwarf-two.o'
Target:    'codeview-two.o'

Logical View:
    {File} 'dwarf-two.o'
      {CompileUnit} 'test.cpp'
-1      {Function} extern not_inlined 'bar' -> 'void'
-         {Parameter} '' -> 'int'
-3      {Function} extern not_inlined 'foo' -> 'void'
-         {CallSite} 'bar' -> 'void'
-           {CallSiteParameter} '' -> 'void'
-           {Parameter} '' -> 'int'
-         {CallSite} 'bar' -> 'void'
-           {CallSiteParameter} '' -> 'void'
-           {Parameter} '' -> 'int'
-         {Block}
-4          {Variable} 'Index' -> 'int'
 5          {Line}
 5          {Line}
 5          {Line}
 3        {Line}
+       {Function} extern not_inlined 'foo' -> 'void'
          {Block}
 5          {Line}
 3        {Line}
```

Compare: view layout

Summary

Debug Information

- Common problems
- LLVM and debug information

llvm-debuginfo-analyzer

- Command line tool that processes debug information
- Produces a uniform logical view regardless of the encoding of the debug information
- Uses a free form text output for the logical view
- Prints the logical elements representing the debug information
- Supports selection criteria to determine logical elements to print
- Finds semantic differences by comparing logical views

C. Future work

- Add support for binary formats:
  - WebAssembly (Wasm)
  - Extended COFF (XCOFF)
- Generate the logical views in:
  - JSON or YAML
- Process additional debug information data:
  - DWARF v5 .debug_names section
  - CodeView public symbols stream
- Support relocatable files:
  - Ability to process objects where each function is always in a different section (deadstripping).

Big thank you

- Paul Robinson
- David Blaikie
- Jeremy Morse
- Stephen Tozer
- Wolfgang Pieb
- J. Ryan Stinnett
- Djordje Todorovic
- Russell Gallop
- Zequan Wu
- Michał Górny
- Kevin Athey
- Tobias Hieta
- Jonas Devlieghere
- Pavel Samolysov
- Orlando Cazalet-Hyams
- Chris Jackson
- Greg Bedwell
- Eric Christopher
- Reid Kleckner
- Alexandre Ganea
- Douglas Yung
- Nico Weber
- Fangrui Song
- Vitaly Buka
- Heejin Ahn
- Adrian Prantl

Thank you!