Assignment 2: Dataflow Framework

Generated by Doxygen 1.8.17

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 anonymous_namespace{available.cpp} Namespace Reference

Classes

- class AvailableExpressions

  Primary function pass to run AvailableExpressions pass.
- class KillGenEval

  KillGenEval is a subclass of KillGen class, which is a template class. Main function of this class is to provide a killEval and genEval function to take in the input bit set which is the result of the meet operator, the current Basic Block, and the domainset, which is a vector of objects we want to perform the analysis on (Expression in this case) and returns the resultant kill set or gen set bits respectively.

Functions

- RegisterPass< AvailableExpressions > X ("available", "ECE 5984 Available Expressions")

5.1.1 Function Documentation

5.1.1.1 X()

```
RegisterPass<AvailableExpressions> anonymous_namespace{available.cpp}::X (
             "available" ,
             "ECE 5984 Available Expressions"  )
```

5.2 anonymous_namespace{liveness.cpp} Namespace Reference

Classes

- class KillGenLive

  KillGenLive is a subclass of KillGen class, which is a template class. Main function of this class is to provide a killEval and genEval function to take in the input bit set which is the result of the meet operator, the current Basic Block, and the domainset, which is a vector of objects we want to perform the analysis on (variables/Value type in thie case) and returns the resultant kill set or gen set bits respectively.
- class Liveness

Functions

- RegisterPass< Liveness > X ("liveness", "ECE 5984 Liveness")

5.2.1 Function Documentation

5.2.1.1 X()

```
RegisterPass<Liveness> anonymous_namespace{liveness.cpp}::X (
             "liveness" ,
             "ECE 5984 Liveness"  )
```

5.3 llvm Namespace Reference

Classes

- class BaseTransferFunction

  Holds the base implementation of a transfer function, to be extended later if we require additional steps to be added to the transfer function. Currently the only method, which is called run takes in the input, genset, and killset, and returns the result of [Gen U (In - Kill)].
- class BBInOutBits

  Holds the bitsets for each basic block's IN and OUT. Owner of the memory.
- class DataflowFramework

  Primary Dataflow Framework template class. Performs the generalized steps of initializing the IN and OUT, calling the Gen and Kill functions and then passing the results to the Transfer Function. Result of which gets set/cleared in the IN/OUT of the correct BB.
- class Expression
- class KillGen

  Interface class for the Kill and Gen functionality. Any new data flow framework must implement their own killEval and genEval functions as they each operate on their domain in their own ways.

Functions

- void printSet (std::vector< Expression > ∗x)
- std::string getShortValueName (Value ∗v)

5.3.1 Function Documentation

5.3.1.1 getShortValueName()

```
std::string llvm::getShortValueName (
             Value * v )
```

5.3.1.2 printSet()

```
void llvm::printSet (
             std::vector< Expression > * x )
```

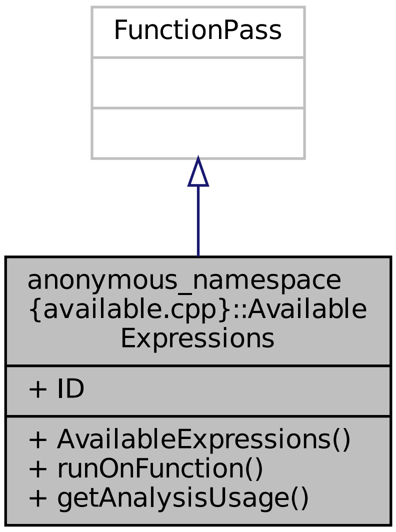
Chapter 6

Class Documentation

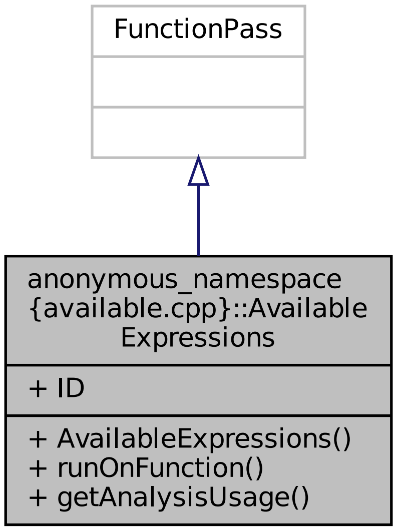
6.1 anonymous_namespace{available.cpp}::AvailableExpressions Class Reference

Primary function pass to run AvailableExpressions pass.

Inheritance diagram for anonymous_namespace{available.cpp}::AvailableExpressions:



Collaboration diagram for anonymous_namespace{available.cpp}::AvailableExpressions:



Public Member Functions

- AvailableExpressions ()
- virtual bool runOnFunction (Function &F)
- virtual void getAnalysisUsage (AnalysisUsage &AU) const

Static Public Attributes

- static char ID = 0

6.1.1 Detailed Description

Primary function pass to run AvailableExpressions pass.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 AvailableExpressions()

```
anonymous_namespace{available.cpp}::AvailableExpressions::AvailableExpressions ( )  [inline]
```

6.1.3 Member Function Documentation

6.1.3.1 getAnalysisUsage()

```
virtual void anonymous_namespace{available.cpp}::AvailableExpressions::getAnalysisUsage (
            AnalysisUsage & AU ) const  [inline], [virtual]
```

6.1.3.2 runOnFunction()

```
virtual bool anonymous_namespace{available.cpp}::AvailableExpressions::runOnFunction (
            Function & F )  [inline], [virtual]
```

6.1.4 Member Data Documentation

6.1.4.1 ID

```
char anonymous_namespace{available.cpp}::AvailableExpressions::ID = 0  [static]
```

The documentation for this class was generated from the following file:

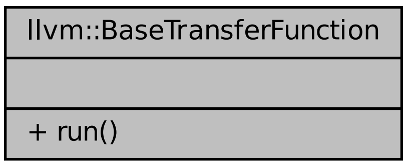
- available.cpp

6.2 llvm::BaseTransferFunction Class Reference

Holds the base implementation of a transfer function, to be extended later if we require additional steps to be added to the transfer function. Currently the only method, which is called run takes in the input, genset, and killset, and returns the result of [Gen U (In - Kill)].

```
#include <BaseTransferFunction.h>
```

Collaboration diagram for llvm::BaseTransferFunction:



Public Member Functions

- virtual std::bitset< MAX_BITS_SIZE > run (const std::bitset< MAX_BITS_SIZE > &input, const std::bitset< MAX_BITS_SIZE > &genSet, const std::bitset< MAX_BITS_SIZE > &killSet)

  General base transfer function main method. Takes in 3 const set references, gen, kill, and input(IN for forward, OUT for backward analysis) General form is [Gen U (In - Kill)]. To mimic the (In - Kill) without doing a borrow operation, we flip the kill set and perform bitwise AND. Truth table is 0-0 = 0; 0-1 = 0; 1-0 = 1; 1-1 = 0. Next, Union operation is synonymous to bitwise OR. Truth table is 0 U 0 = 0; 0 U 1 = 1; 1 U 0 = 1; 1 U 1 = 1.

6.2.1 Detailed Description

Holds the base implementation of a transfer function, to be extended later if we require additional steps to be added to the transfer function. Currently the only method, which is called run takes in the input, genset, and killset, and returns the result of [Gen U (In - Kill)].

6.2.2 Member Function Documentation

6.2.2.1 run()

```
std::bitset< MAX_BITS_SIZE > llvm::BaseTransferFunction::run (
            const std::bitset< MAX_BITS_SIZE > & input,
            const std::bitset< MAX_BITS_SIZE > & genSet,
            const std::bitset< MAX_BITS_SIZE > & killSet )  [virtual]
```

General base transfer function main method. Takes in 3 const set references, gen, kill, and input(IN for forward, OUT for backward analysis) General form is [Gen U (In - Kill)]. To mimic the (In - Kill) without doing a borrow operation, we flip the kill set and perform bitwise AND. Truth table is 0-0 = 0; 0-1 = 0; 1-0 = 1; 1-1 = 0. Next, Union operation is synonymous to bitwise OR. Truth table is 0 U 0 = 0; 0 U 1 = 1; 1 U 0 = 1; 1 U 1 = 1.

Parameters

input	Input into function, can be IN or OUT depending on direction
genSet	Gen set
killSet	Kill set

Returns

Copy out the bitset.

The documentation for this class was generated from the following files:

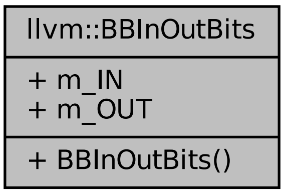
- DataflowFramework/include/BaseTransferFunction.h
- DataflowFramework/BaseTransferFunction.cpp

6.3 llvm::BBInOutBits Class Reference

Holds the bitsets for each basic block's IN and OUT. Owner of the memory.

```
#include <dataflow.h>
```

Collaboration diagram for llvm::BBInOutBits:



Public Member Functions

- BBInOutBits (BitsVal inval, BitsVal outval)

  Overload ctor takes inval/outval as ZEROS or ONES and initializes the IN/OUT with the corresponding value.

Public Attributes

- std::bitset< MAX_BITS_SIZE > m_IN
- std::bitset< MAX_BITS_SIZE > m_OUT

6.3.1 Detailed Description

Holds the bitsets for each basic block's IN and OUT. Owner of the memory.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 BBInOutBits()

```
llvm::BBInOutBits::BBInOutBits (
            BitsVal inval,
            BitsVal outval )  [inline]
```

Overload ctor takes inval/outval as ZEROS or ONES and initializes the IN/OUT with the corresponding value.

Parameters

inval	
outval	

6.3.3 Member Data Documentation

6.3.3.1 m_IN

```
std::bitset<MAX_BITS_SIZE> llvm::BBInOutBits::m_IN
```

6.3.3.2 m_OUT

```
std::bitset<MAX_BITS_SIZE> llvm::BBInOutBits::m_OUT
```

The documentation for this class was generated from the following file:

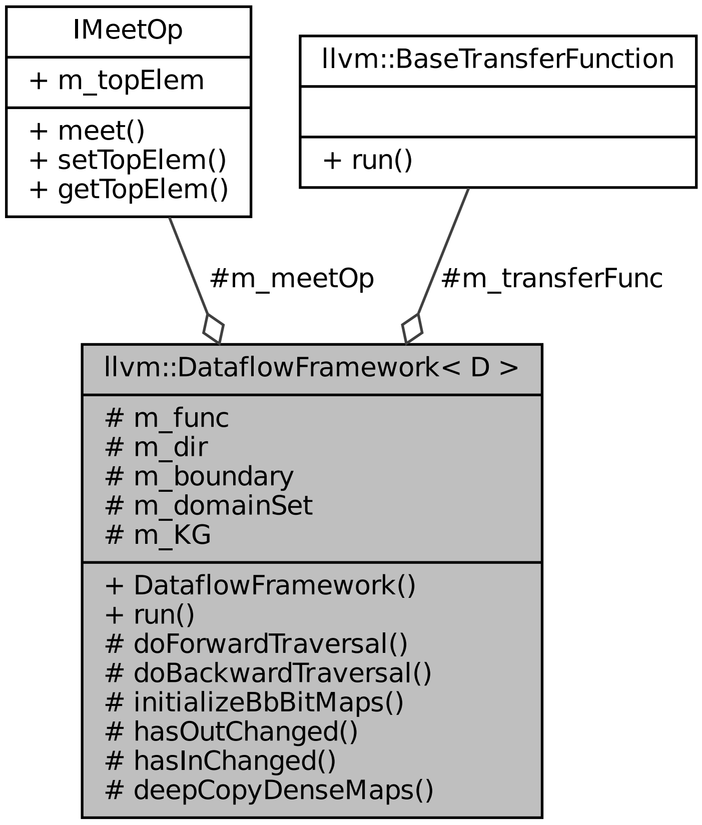
- DataflowFramework/include/dataflow.h

6.4 llvm::DataflowFramework< D > Class Template Reference

Primary Dataflow Framework template class. Performs the generalized steps of initializing the IN and OUT, calling the Gen and Kill functions and then passing the results to the Transfer Function. Result of which gets set/cleared in the IN/OUT of the correct BB.

```
#include <dataflow.h>
```

Collaboration diagram for llvm::DataflowFramework< D >:



Public Member Functions

- DataflowFramework (IMeetOp &meetOp, FlowDirection direction, BoundaryCondition boundary, Function &function, std::vector< D > &domainset, KillGen< D > &KillGenImp, BaseTransferFunction &transfer)
- std::vector< D > & run ()

  Primary run function of the Dataflow Framework.

Protected Member Functions

- void doForwardTraversal (llvm::DenseMap< BasicBlock ∗, BBInOutBits ∗ > &currentInOutMap, llvm::↩ DenseMap< BasicBlock ∗, BBInOutBits ∗ > &previousInOutMap)

  Primary function for forward traversal. Iterates through basic blocks in an Inverse Post Order direction.
- void doBackwardTraversal (llvm::DenseMap< BasicBlock ∗, BBInOutBits ∗ > &currentInOutMap, llvm::↩ DenseMap< BasicBlock ∗, BBInOutBits ∗ > &previousInOutMap)

  Primary function for backward traversal. Iterates through basic blocks in a Post Order direction.
- void initializeBbBitMaps (Function &F, llvm::DenseMap< BasicBlock ∗, BBInOutBits ∗ > &map)

  Initializes the basic block bitmaps for in and out. In charge of creating the bit vectors and associating them with the basic blocks.

- bool hasOutChanged (llvm::DenseMap< BasicBlock ∗, BBInOutBits ∗ > &currentMap, llvm::DenseMap< BasicBlock ∗, BBInOutBits ∗ > &previousMap)

  Checks if any of the OUT's of any basic blocks has changed, if it has, return true, else return false.
- bool hasInChanged (llvm::DenseMap< BasicBlock ∗, BBInOutBits ∗ > &currentMap, llvm::DenseMap< BasicBlock ∗, BBInOutBits ∗ > &previousMap)

  Checks if any of the IN's of any basic blocks has changed, if it has, return true, else return false.
- void deepCopyDenseMaps (llvm::DenseMap< BasicBlock ∗, BBInOutBits ∗ > &currentMap, llvm::Dense↩ Map< BasicBlock ∗, BBInOutBits ∗ > &previousMap)

  Creates copies the memory contents of IN and OUT from currentMap into previousMap.

Protected Attributes

- IMeetOp & m_meetOp
- Function & m_func
- FlowDirection m_dir
- BoundaryCondition m_boundary
- std::vector< D > & m_domainSet
- KillGen< D > & m_KG
- BaseTransferFunction & m_transferFunc

6.4.1 Detailed Description

template<typename D>
class llvm::DataflowFramework< D >

Primary Dataflow Framework template class. Performs the generalized steps of initializing the IN and OUT, calling the Gen and Kill functions and then passing the results to the Transfer Function. Result of which gets set/cleared in the IN/OUT of the correct BB.

Template Parameters

D	Domain we operate on. (Values, Expressions, etc.)

6.4.2 Constructor & Destructor Documentation

6.4.2.1 DataflowFramework()

```
template<typename D >
llvm::DataflowFramework< D >::DataflowFramework (
            IMeetOp & meetOp,
            FlowDirection direction,
            BoundaryCondition boundary,
            Function & function,
            std::vector< D > & domainset,
            KillGen< D > & KillGenImp,
            BaseTransferFunction & transfer )
```

6.4.3 Member Function Documentation

6.4.3.1 deepCopyDenseMaps()

```
template<typename D >
void llvm::DataflowFramework< D >::deepCopyDenseMaps (
             llvm::DenseMap< BasicBlock *, BBInOutBits * > & currentMap,
             llvm::DenseMap< BasicBlock *, BBInOutBits * > & previousMap )  [protected]
```

Creates copies the memory contents of IN and OUT from currentMap into previousMap.

Template Parameters

D	

Parameters

currentMap	
previousMap	

6.4.3.2 doBackwardTraversal()

```
template<typename D >
void llvm::DataflowFramework< D >::doBackwardTraversal (
             llvm::DenseMap< BasicBlock *, BBInOutBits * > & currentInOutMap,
             llvm::DenseMap< BasicBlock *, BBInOutBits * > & previousInOutMap )  [protected]
```

Primary function for backward traversal. Iterates through basic blocks in a Post Order direction.

Template Parameters

D	

Parameters

currentInOutMap	
previousInOutMap	

6.4.3.3 doForwardTraversal()

```
template<typename D >
void llvm::DataflowFramework< D >::doForwardTraversal (
             llvm::DenseMap< BasicBlock *, BBInOutBits * > & currentInOutMap,
             llvm::DenseMap< BasicBlock *, BBInOutBits * > & previousInOutMap )  [protected]
```

Primary function for forward traversal. Iterates through basic blocks in an Inverse Post Order direction.

Template Parameters

D	

Parameters

currentInOutMap	
previousInOutMap	

6.4.3.4 hasInChanged()

```
template<typename D >
bool llvm::DataflowFramework< D >::hasInChanged (
             llvm::DenseMap< BasicBlock *, BBInOutBits * > & currentMap,
             llvm::DenseMap< BasicBlock *, BBInOutBits * > & previousMap )  [protected]
```

Checks if any of the IN's of any basic blocks has changed, if it has, return true, else return false.

Template Parameters

D	Domain we operate on

Parameters

currentMap	Current bitmap reference
previousMap	Previous bitmap reference from previous iteration

Returns

6.4.3.5 hasOutChanged()

```
template<typename D >
bool llvm::DataflowFramework< D >::hasOutChanged (
             llvm::DenseMap< BasicBlock *, BBInOutBits * > & currentMap,
             llvm::DenseMap< BasicBlock *, BBInOutBits * > & previousMap )  [protected]
```

Checks if any of the OUT's of any basic blocks has changed, if it has, return true, else return false.

Template Parameters

D	Domain we operate on

Parameters

currentMap	Current bitmap reference
previousMap	Previous bitmap reference from previous iteration

Returns

6.4.3.6 initializeBbBitMaps()

```
template<typename D >
void llvm::DataflowFramework< D >::initializeBbBitMaps (
             Function & F,
             llvm::DenseMap< BasicBlock *, BBInOutBits * > & map )  [protected]
```

Initializes the basic block bitmaps for in and out. In charge of creating the bit vectors and associating them with the basic blocks.

Template Parameters

D	Domain we operate on

Parameters

F	Function reference we're operating on
currentMap	Map reference for basic block pointer to IN OUT bitmap mapping

6.4.3.7 run()

```
template<typename D >
std::vector< D > & llvm::DataflowFramework< D >::run
```

Primary run function of the Dataflow Framework.

Template Parameters

D	

Returns

6.4.4 Member Data Documentation

6.4.4.1 m_boundary

```
template<typename D >
BoundaryCondition llvm::DataflowFramework< D >::m_boundary  [protected]
```

6.4.4.2 m_dir

```
template<typename D >
FlowDirection llvm::DataflowFramework< D >::m_dir  [protected]
```

6.4.4.3 m_domainSet

```
template<typename D >
std::vector<D>& llvm::DataflowFramework< D >::m_domainSet  [protected]
```

6.4.4.4 m_func

```
template<typename D >
Function& llvm::DataflowFramework< D >::m_func  [protected]
```

6.4.4.5 m_KG

```
template<typename D >
KillGen<D>& llvm::DataflowFramework< D >::m_KG  [protected]
```

6.4.4.6 m_meetOp

```
template<typename D >
IMeetOp& llvm::DataflowFramework< D >::m_meetOp  [protected]
```

6.4.4.7 m_transferFunc

```
template<typename D >
BaseTransferFunction& llvm::DataflowFramework< D >::m_transferFunc  [protected]
```

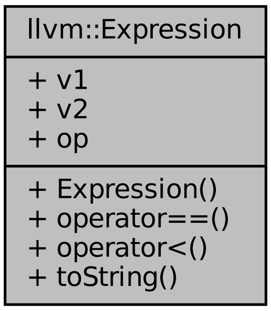
The documentation for this class was generated from the following file:

- DataflowFramework/include/dataflow.h

6.5 llvm::Expression Class Reference

```
#include <available-support.h>
```

Collaboration diagram for llvm::Expression:



Public Member Functions

- Expression (Instruction ∗I)
- bool operator== (const Expression &e2) const
- bool operator< (const Expression &e2) const
- std::string toString () const

Public Attributes

- Value ∗ v1
- Value ∗ v2
- Instruction::BinaryOps op

6.5.1 Constructor & Destructor Documentation

6.5.1.1 Expression()

```
llvm::Expression::Expression (
             Instruction * I )
```

6.5.2 Member Function Documentation

6.5.2.1 operator<()

```
bool llvm::Expression::operator< (
             const Expression & e2 ) const
```

6.5.2.2 operator==()

```
bool llvm::Expression::operator== (
             const Expression & e2 ) const
```

6.5.2.3 toString()

```
std::string llvm::Expression::toString ( ) const
```

6.5.3 Member Data Documentation

6.5.3.1 op

```
Instruction::BinaryOps llvm::Expression::op
```

6.5.3.2 v1

```
Value* llvm::Expression::v1
```

6.5.3.3 v2

```
Value* llvm::Expression::v2
```

The documentation for this class was generated from the following files:

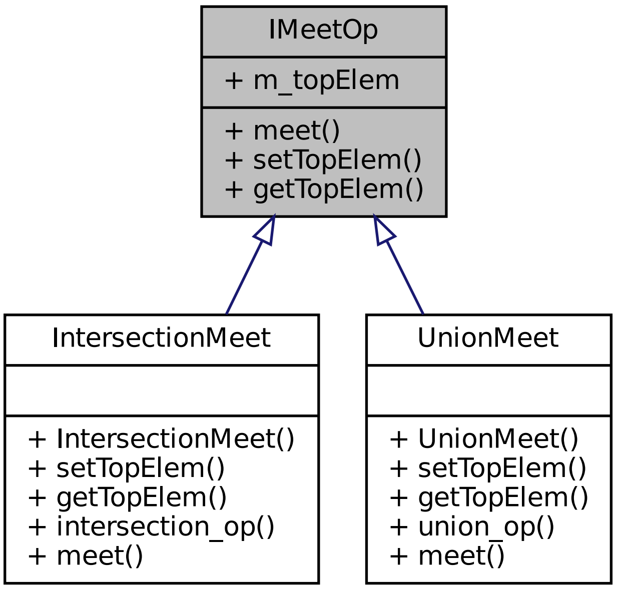
- available-support.h
- available-support.cpp

6.6 IMeetOp Class Reference

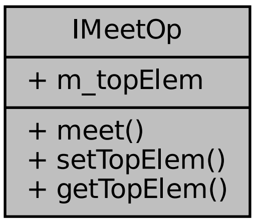
MeetOperator pure virtual class. Any new meet operator to be added must inherit this class and implement their own meet function and getters/setters for the top element.

```
#include <MeetOpInterface.h>
```

Inheritance diagram for IMeetOp:



Collaboration diagram for IMeetOp:



Public Member Functions

- virtual std::bitset< MAX_BITS_SIZE > meet (std::bitset< MAX_BITS_SIZE > input1, std::bitset< MAX_BITS_SIZE > input2)=0
- virtual void setTopElem (BitsVal val)=0
- virtual BitsVal getTopElem ()=0

Public Attributes

- BitsVal m_topElem

6.6.1 Detailed Description

MeetOperator pure virtual class. Any new meet operator to be added must inherit this class and implement their own meet function and getters/setters for the top element.

6.6.2 Member Function Documentation

6.6.2.1 getTopElem()

```
virtual BitsVal IMeetOp::getTopElem ( )  [pure virtual]
```

Implemented in IntersectionMeet, and UnionMeet.

6.6.2.2 meet()

```
virtual std::bitset<MAX_BITS_SIZE> IMeetOp::meet (
            std::bitset< MAX_BITS_SIZE > input1,
            std::bitset< MAX_BITS_SIZE > input2 )  [pure virtual]
```

Implemented in IntersectionMeet, and UnionMeet.

6.6.2.3 setTopElem()

```
virtual void IMeetOp::setTopElem (
            BitsVal val )  [pure virtual]
```

Implemented in IntersectionMeet, and UnionMeet.

6.6.3 Member Data Documentation

6.6.3.1 m_topElem

```
BitsVal IMeetOp::m_topElem
```

The documentation for this class was generated from the following file:

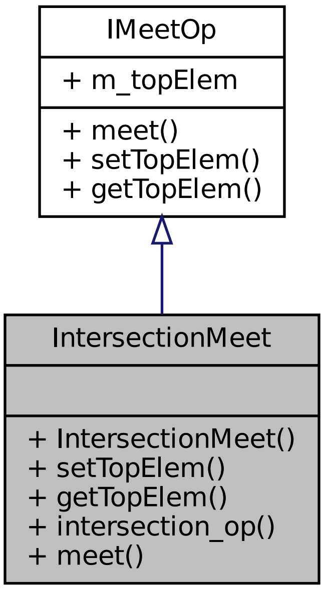
- DataflowFramework/include/MeetOpInterface.h

6.7 IntersectionMeet Class Reference

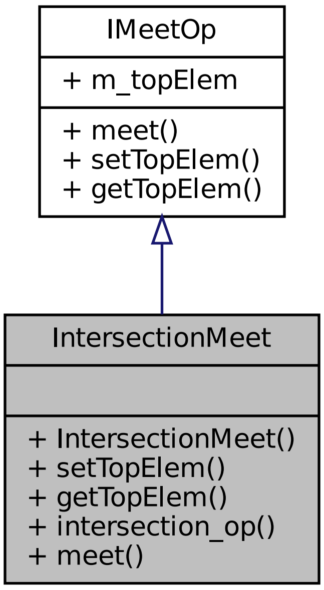
Intersection meet, implements the meet operation, and is able to set and get the top element.

```
#include <IntersectionMeet.h>
```

Inheritance diagram for IntersectionMeet:



Collaboration diagram for IntersectionMeet:



Public Member Functions

- IntersectionMeet ()
- void setTopElem (BitsVal val) override
- BitsVal getTopElem () override
- std::bitset< MAX_BITS_SIZE > intersection_op (std::bitset< MAX_BITS_SIZE > ip1, std::bitset< MAX_BITS_SIZE > ip2)
- std::bitset< MAX_BITS_SIZE > meet (std::bitset< MAX_BITS_SIZE > input1, std::bitset< MAX_BITS_SIZE > input2) override

Additional Inherited Members

6.7.1 Detailed Description

Intersection meet, implements the meet operation, and is able to set and get the top element.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 IntersectionMeet()

```
IntersectionMeet::IntersectionMeet ( )
```

6.7.3 Member Function Documentation

6.7.3.1 getTopElem()

```
BitsVal IntersectionMeet::getTopElem ( )  [override], [virtual]
```

Implements IMeetOp.

6.7.3.2 intersection_op()

```
std::bitset< MAX_BITS_SIZE > IntersectionMeet::intersection_op (
            std::bitset< MAX_BITS_SIZE > ip1,
            std::bitset< MAX_BITS_SIZE > ip2 )
```

6.7.3.3 meet()

```
std::bitset< MAX_BITS_SIZE > IntersectionMeet::meet (
            std::bitset< MAX_BITS_SIZE > input1,
            std::bitset< MAX_BITS_SIZE > input2 )  [override], [virtual]
```

Implements IMeetOp.

6.7.3.4 setTopElem()

```
void IntersectionMeet::setTopElem (
            BitsVal val )  [override], [virtual]
```

Implements IMeetOp.

The documentation for this class was generated from the following files:

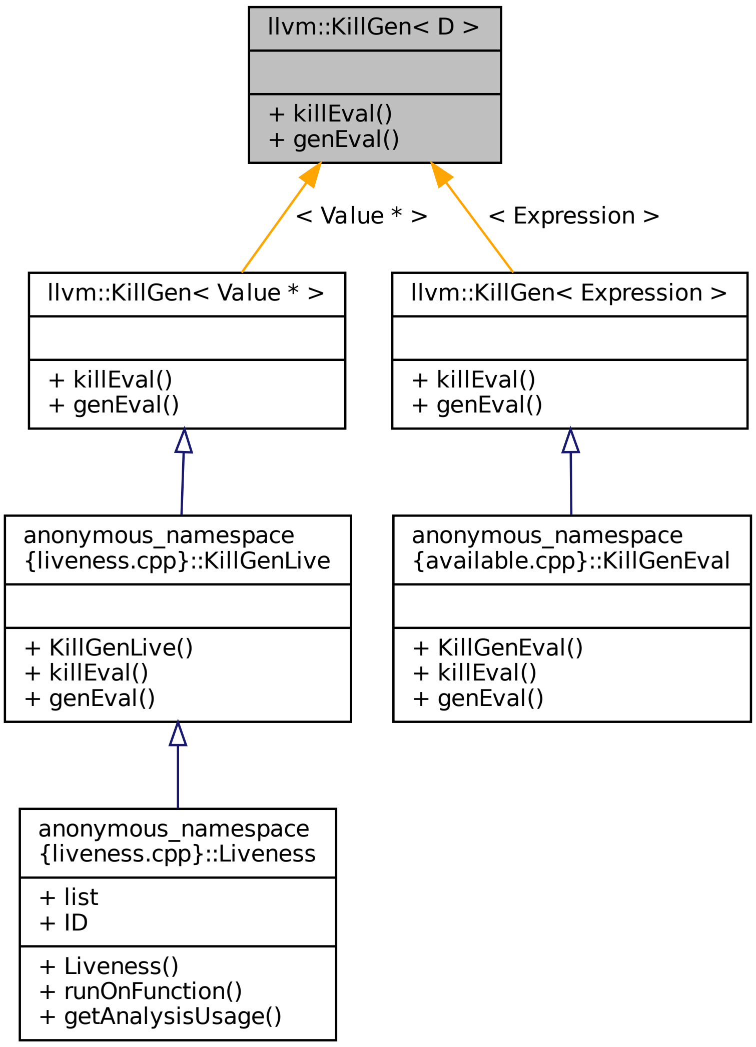
- DataflowFramework/include/IntersectionMeet.h
- DataflowFramework/IntersectionMeet.cpp

6.8 llvm::KillGen< D > Class Template Reference

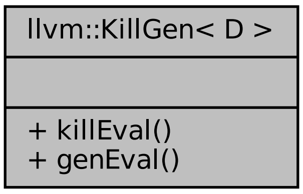
Interface class for the Kill and Gen functionality. Any new data flow framework must implement their own killEval and genEval functions as they each operate on their domain in their own ways.

```
#include <KillGen.h>
```

Inheritance diagram for llvm::KillGen< D >:



Collaboration diagram for llvm::KillGen< D >:



Public Member Functions

- virtual std::bitset< MAX_BITS_SIZE > killEval (llvm::BasicBlock ∗BB, std::bitset< MAX_BITS_SIZE > &meet_res, std::vector< D > &domainset)=0
- virtual std::bitset< MAX_BITS_SIZE > genEval (llvm::BasicBlock ∗BB, std::bitset< MAX_BITS_SIZE > &meet_res, std::vector< D > &domainset)=0

6.8.1 Detailed Description

template<typename D>
class llvm::KillGen< D >

Interface class for the Kill and Gen functionality. Any new data flow framework must implement their own killEval and genEval functions as they each operate on their domain in their own ways.

Template Parameters

D	Domain we operate on (variables/values or expressions)

6.8.2 Member Function Documentation

6.8.2.1 genEval()

```
template<typename D >
virtual std::bitset<MAX_BITS_SIZE> llvm::KillGen< D >::genEval (
             llvm::BasicBlock * BB,
             std::bitset< MAX_BITS_SIZE > & meet_res,
             std::vector< D > & domainset )  [pure virtual]
```

Implemented in anonymous_namespace{liveness.cpp}::KillGenLive, and anonymous_namespace{available.cpp}::KillGenEval.

6.8.2.2 killEval()

```
template<typename D >
virtual std::bitset<MAX_BITS_SIZE> llvm::KillGen< D >::killEval (
             llvm::BasicBlock * BB,
             std::bitset< MAX_BITS_SIZE > & meet_res,
             std::vector< D > & domainset )  [pure virtual]
```

Implemented in anonymous_namespace{liveness.cpp}::KillGenLive, and anonymous_namespace{available.cpp}::KillGenEval.

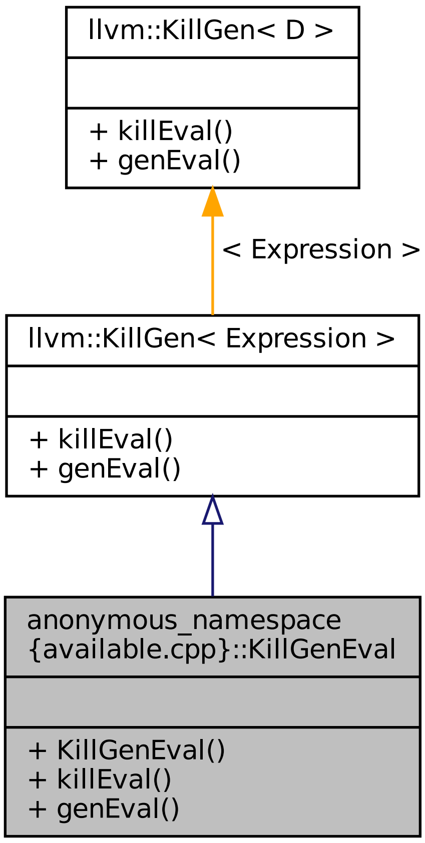
The documentation for this class was generated from the following file:

- DataflowFramework/include/KillGen.h

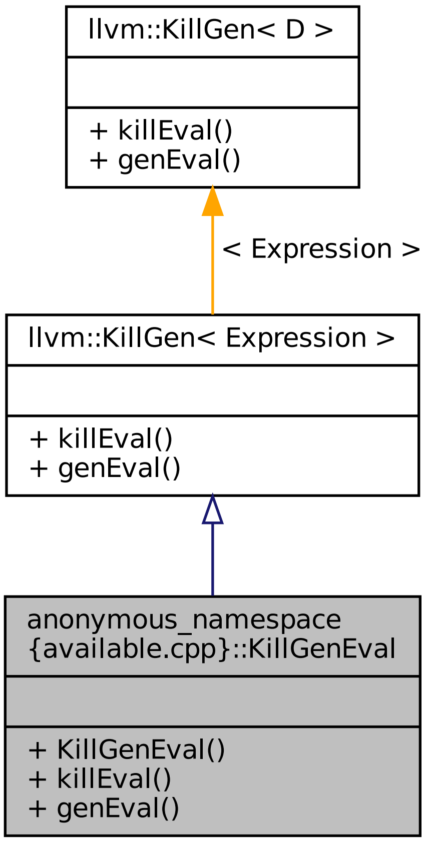
6.9 anonymous_namespace{available.cpp}::KillGenEval Class Reference

KillGenEval is a subclass of KillGen class, which is a template class. Main function of this class is to provide a killEval and genEval function to take in the input bit set which is the result of the meet operator, the current Basic Block, and the domainset, which is a vector of objects we want to perform the analysis on (Expression in this case) and returns the resultant kill set or gen set bits respectively.

Inheritance diagram for anonymous_namespace{available.cpp}::KillGenEval:



Collaboration diagram for anonymous_namespace{available.cpp}::KillGenEval:



Public Member Functions

- KillGenEval ()
- std::bitset< MAX_BITS_SIZE > killEval (llvm::BasicBlock ∗BB, std::bitset< MAX_BITS_SIZE > &meet_res, std::vector< Expression > &domainset) override
- std::bitset< MAX_BITS_SIZE > genEval (llvm::BasicBlock ∗BB, std::bitset< MAX_BITS_SIZE > &meet_res, std::vector< Expression > &domainset) override

6.9.1 Detailed Description

KillGenEval is a subclass of KillGen class, which is a template class. Main function of this class is to provide a killEval and genEval function to take in the input bit set which is the result of the meet operator, the current Basic Block, and the domainset, which is a vector of objects we want to perform the analysis on (Expression in this case) and returns the resultant kill set or gen set bits respectively.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 KillGenEval()

```
anonymous_namespace{available.cpp}::KillGenEval::KillGenEval ( )  [inline]
```

6.9.3 Member Function Documentation

6.9.3.1 genEval()

```
std::bitset<MAX_BITS_SIZE> anonymous_namespace{available.cpp}::KillGenEval::genEval (
            llvm::BasicBlock * BB,
            std::bitset< MAX_BITS_SIZE > & meet_res,
            std::vector< Expression > & domainset )  [inline], [override], [virtual]
```

Implements llvm::KillGen< Expression >.

6.9.3.2 killEval()

```
std::bitset<MAX_BITS_SIZE> anonymous_namespace{available.cpp}::KillGenEval::killEval (
            llvm::BasicBlock * BB,
            std::bitset< MAX_BITS_SIZE > & meet_res,
            std::vector< Expression > & domainset )  [inline], [override], [virtual]
```

Implements llvm::KillGen< Expression >.

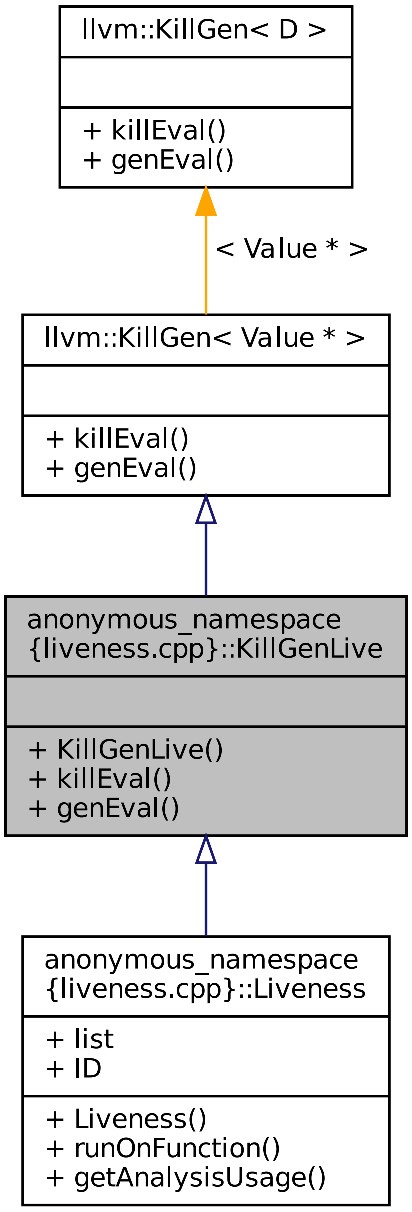
The documentation for this class was generated from the following file:

- available.cpp

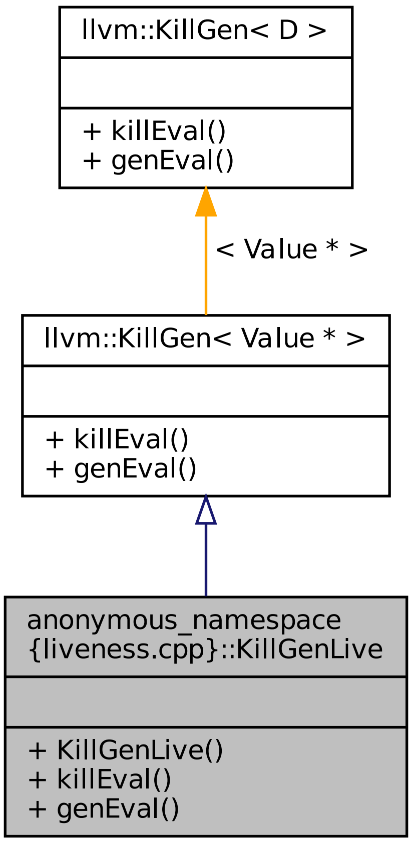
6.10 anonymous_namespace{liveness.cpp}::KillGenLive Class Reference

KillGenLive is a subclass of KillGen class, which is a template class. Main function of this class is to provide a killEval and genEval function to take in the input bit set which is the result of the meet operator, the current Basic Block, and the domainset, which is a vector of objects we want to perform the analysis on (variables/Value type in thie case) and returns the resultant kill set or gen set bits respectively.

Inheritance diagram for anonymous_namespace{liveness.cpp}::KillGenLive:



Collaboration diagram for anonymous_namespace{liveness.cpp}::KillGenLive:



Public Member Functions

- KillGenLive ()
- std::bitset< MAX_BITS_SIZE > killEval (llvm::BasicBlock ∗BB, std::bitset< MAX_BITS_SIZE > &meet_res, std::vector< Value ∗ > &domainset) override
- std::bitset< MAX_BITS_SIZE > genEval (llvm::BasicBlock ∗BB, std::bitset< MAX_BITS_SIZE > &meet_res, std::vector< Value ∗ > &domainset) override

6.10.1 Detailed Description

KillGenLive is a subclass of KillGen class, which is a template class. Main function of this class is to provide a killEval and genEval function to take in the input bit set which is the result of the meet operator, the current Basic Block, and the domainset, which is a vector of objects we want to perform the analysis on (variables/Value type in thie case) and returns the resultant kill set or gen set bits respectively.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 KillGenLive()

```
anonymous_namespace{liveness.cpp}::KillGenLive::KillGenLive ( )  [inline]
```

6.10.3 Member Function Documentation

6.10.3.1 genEval()

```
std::bitset<MAX_BITS_SIZE> anonymous_namespace{liveness.cpp}::KillGenLive::genEval (
            llvm::BasicBlock * BB,
            std::bitset< MAX_BITS_SIZE > & meet_res,
            std::vector< Value * > & domainset )  [inline], [override], [virtual]
```

Implements llvm::KillGen< Value * >.

6.10.3.2 killEval()

```
std::bitset<MAX_BITS_SIZE> anonymous_namespace{liveness.cpp}::KillGenLive::killEval (
            llvm::BasicBlock * BB,
            std::bitset< MAX_BITS_SIZE > & meet_res,
            std::vector< Value * > & domainset )  [inline], [override], [virtual]
```

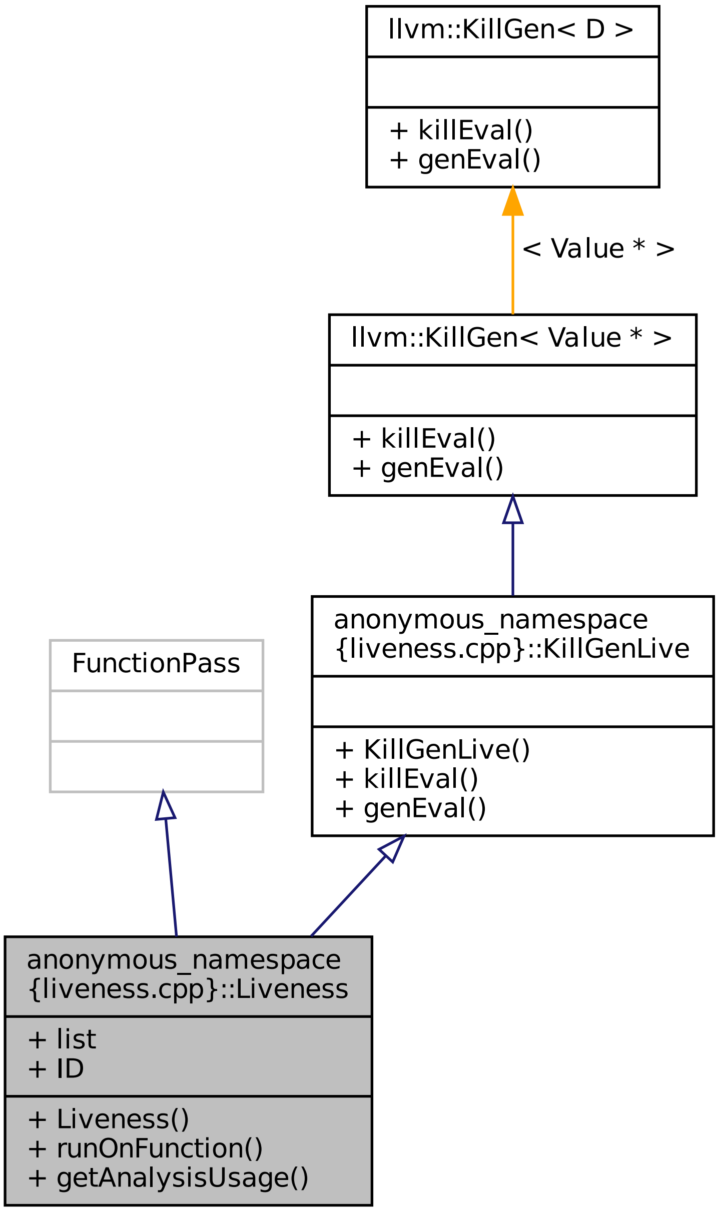
Implements llvm::KillGen< Value * >.

The documentation for this class was generated from the following file:

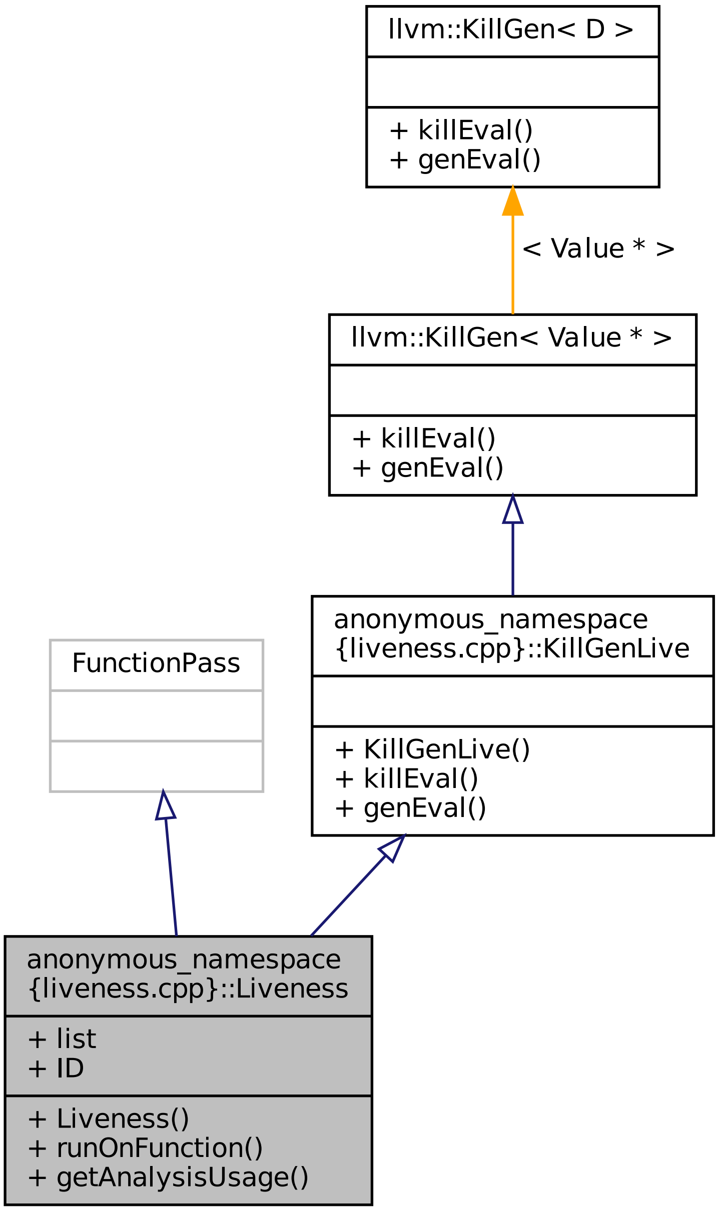
- liveness.cpp

6.11 anonymous_namespace{liveness.cpp}::Liveness Class Reference

Inheritance diagram for anonymous_namespace{liveness.cpp}::Liveness:



Collaboration diagram for anonymous_namespace{liveness.cpp}::Liveness:



Public Member Functions

- Liveness ()
- virtual bool runOnFunction (Function &F)
- virtual void getAnalysisUsage (AnalysisUsage &AU) const

Public Attributes

- std::vector$<$ Value $*$ $>$ list

Static Public Attributes

- static char ID = 0

6.11.1 Constructor & Destructor Documentation

6.11.1.1 Liveness()

```
anonymous_namespace{liveness.cpp}::Liveness::Liveness ( )  [inline]
```

6.11.2 Member Function Documentation

6.11.2.1 getAnalysisUsage()

```
virtual void anonymous_namespace{liveness.cpp}::Liveness::getAnalysisUsage (
             AnalysisUsage & AU ) const  [inline], [virtual]
```

6.11.2.2 runOnFunction()

```
virtual bool anonymous_namespace{liveness.cpp}::Liveness::runOnFunction (
             Function & F )  [inline], [virtual]
```

6.11.3 Member Data Documentation

6.11.3.1 ID

```
char anonymous_namespace{liveness.cpp}::Liveness::ID = 0  [static]
```

6.11.3.2 list

```
std::vector<Value *> anonymous_namespace{liveness.cpp}::Liveness::list
```

The documentation for this class was generated from the following file:

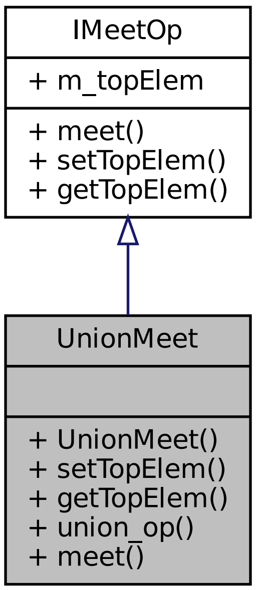
- liveness.cpp

6.12 UnionMeet Class Reference

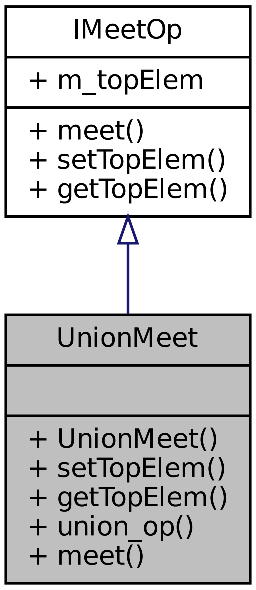
Union meet, implements the meet operation, and is able to set and get the top element.

```
#include <UnionMeet.h>
```

Inheritance diagram for UnionMeet:



Collaboration diagram for UnionMeet:



Public Member Functions

- UnionMeet ()
- void setTopElem (BitsVal val) override
- BitsVal getTopElem () override
- std::bitset< MAX_BITS_SIZE > union_op (std::bitset< MAX_BITS_SIZE > ip1, std::bitset< MAX_BITS_SIZE > ip2)
- std::bitset< MAX_BITS_SIZE > meet (std::bitset< MAX_BITS_SIZE > input1, std::bitset< MAX_BITS_SIZE > input2) override

Additional Inherited Members

6.12.1 Detailed Description

Union meet, implements the meet operation, and is able to set and get the top element.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 UnionMeet()

```
UnionMeet::UnionMeet ( )
```

6.12.3 Member Function Documentation

6.12.3.1 getTopElem()

```
BitsVal UnionMeet::getTopElem ( )  [override], [virtual]
```

Implements IMeetOp.

6.12.3.2 meet()

```
std::bitset< MAX_BITS_SIZE > UnionMeet::meet (
            std::bitset< MAX_BITS_SIZE > input1,
            std::bitset< MAX_BITS_SIZE > input2 )  [override], [virtual]
```

Implements IMeetOp.

6.12.3.3 setTopElem()

```
void UnionMeet::setTopElem (
            BitsVal val )  [override], [virtual]
```

Implements IMeetOp.

6.12.3.4 union_op()

```
std::bitset< MAX_BITS_SIZE > UnionMeet::union_op (
            std::bitset< MAX_BITS_SIZE > ip1,
            std::bitset< MAX_BITS_SIZE > ip2 )
```

The documentation for this class was generated from the following files:

- DataflowFramework/include/UnionMeet.h
- DataflowFramework/UnionMeet.cpp

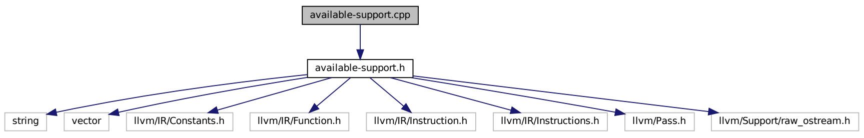
Chapter 7

File Documentation

7.1 available-support.cpp File Reference

```
#include "available-support.h"
```

Include dependency graph for available-support.cpp:



Namespaces

- llvm

Functions

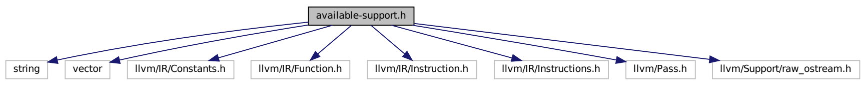
- void llvm::printSet (std::vector< Expression > ∗x)
- std::string llvm::getShortValueName (Value ∗v)

7.2 available-support.h File Reference

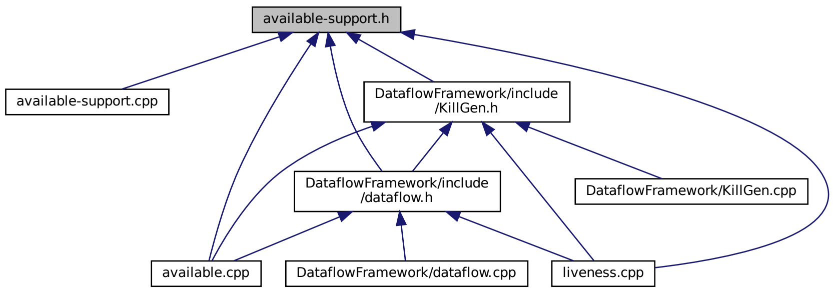
```
#include <string>
#include <vector>
#include "llvm/IR/Constants.h"
#include "llvm/IR/Function.h"
#include "llvm/IR/Instruction.h"
#include "llvm/IR/Instructions.h"
#include "llvm/Pass.h"
```

```
#include "llvm/Support/raw_ostream.h"
```

Include dependency graph for available-support.h:



This graph shows which files directly or indirectly include this file:


available-support.h
available-support.cpp
DataflowFramework/include /KillGen.h
DataflowFramework/include /dataflow.h
DataflowFramework/KillGen.cpp
available.cpp
DataflowFramework/dataflow.cpp
liveness.cpp

Classes

- class llvm::Expression

Namespaces

- llvm

Functions

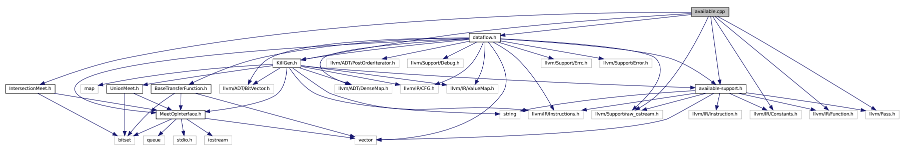
- std::string llvm::getShortValueName (Value $*$v)
- void llvm::printSet (std::vector$<$ Expression $>$ $*$x)

7.3 available.cpp File Reference

```
#include "available-support.h"
#include "llvm/IR/Constants.h"
#include "llvm/IR/Function.h"
#include "llvm/Pass.h"
#include "llvm/Support/raw_ostream.h"
#include <IntersectionMeet.h>
#include <KillGen.h>
```

```
#include <dataflow.h>
```

Include dependency graph for available.cpp:



Classes

- class anonymous_namespace{available.cpp}::KillGenEval

    KillGenEval is a subclass of KillGen class, which is a template class. Main function of this class is to provide a killEval and genEval function to take in the input bit set which is the result of the meet operator, the current Basic Block, and the domainset, which is a vector of objects we want to perform the analysis on (Expression in this case) and returns the resultant kill set or gen set bits respectively.

- class anonymous_namespace{available.cpp}::AvailableExpressions

    Primary function pass to run AvailableExpressions pass.

Namespaces

- anonymous_namespace{available.cpp}

Macros

- #define DEBUG_TYPE "dataflow_framework"

Functions

- RegisterPass< AvailableExpressions > anonymous_namespace{available.cpp}::X ("available", "ECE 5984 Available Expressions")

7.3.1 Macro Definition Documentation

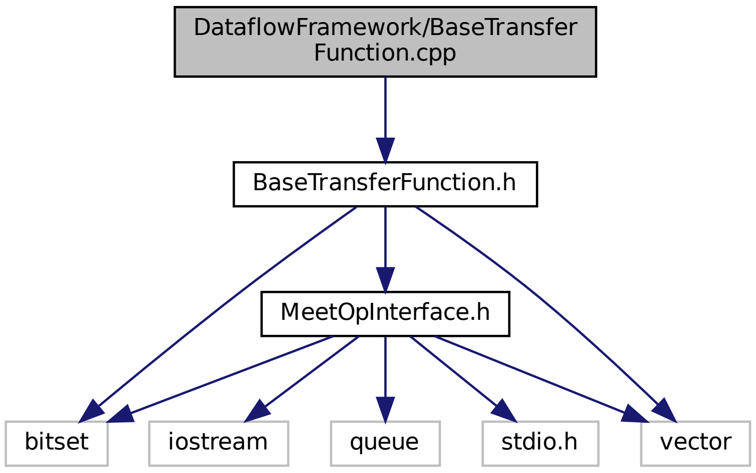
7.3.1.1 DEBUG_TYPE

```
#define DEBUG_TYPE "dataflow_framework"
```

7.4 DataflowFramework/BaseTransferFunction.cpp File Reference

```
#include <BaseTransferFunction.h>
```

Include dependency graph for BaseTransferFunction.cpp:



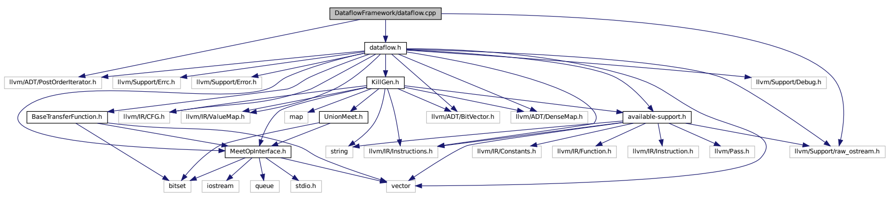
Namespaces

- llvm

7.5 DataflowFramework/dataflow.cpp File Reference

```
#include <dataflow.h>
#include <llvm/ADT/PostOrderIterator.h>
#include <llvm/Support/raw_ostream.h>
```

Include dependency graph for dataflow.cpp:



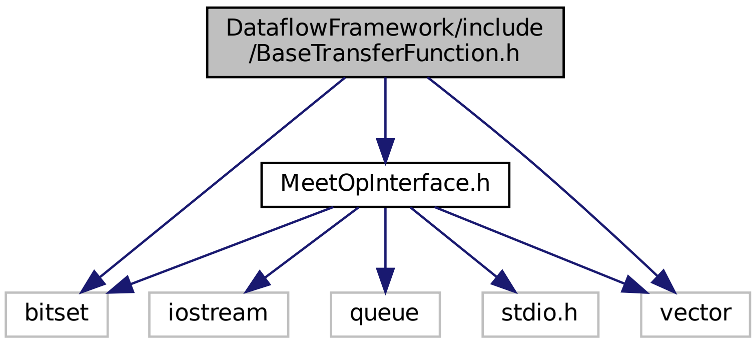
Namespaces

- llvm

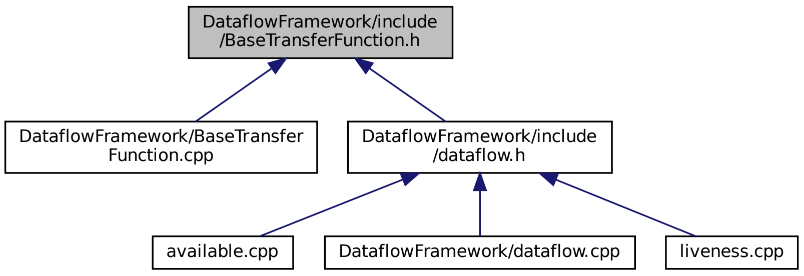
7.6 DataflowFramework/include/BaseTransferFunction.h File Reference

```
#include <MeetOpInterface.h>
#include <bitset>
#include <vector>
```

Include dependency graph for BaseTransferFunction.h:



This graph shows which files directly or indirectly include this file:


DataflowFramework/include
/BaseTransferFunction.h
DataflowFramework/BaseTransfer
Function.cpp
DataflowFramework/include
/dataflow.h
available.cpp
DataflowFramework/dataflow.cpp
liveness.cpp

Classes

- class llvm::BaseTransferFunction

  Holds the base implementation of a transfer function, to be extended later if we require additional steps to be added to the transfer function. Currently the only method, which is called run takes in the input, genset, and killset, and returns the result of [Gen U (In - Kill)].

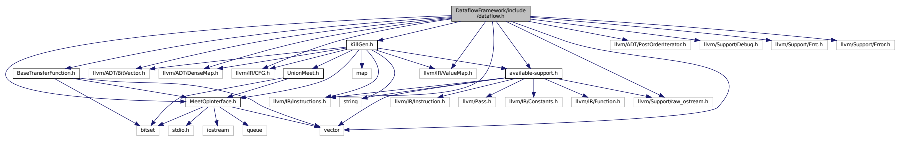
Namespaces

- llvm

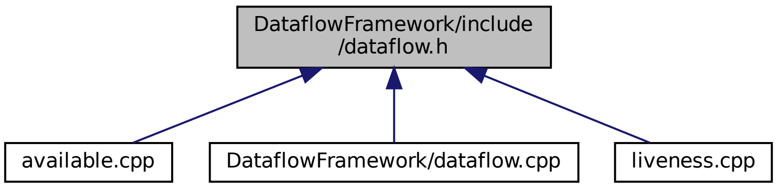
7.7 DataflowFramework/include/dataflow.h File Reference

```
#include "llvm/ADT/BitVector.h"
#include "llvm/ADT/DenseMap.h"
#include "llvm/IR/CFG.h"
#include "llvm/IR/Instructions.h"
#include "llvm/IR/ValueMap.h"
#include <llvm/ADT/PostOrderIterator.h>
#include <llvm/Support/Debug.h>
#include <llvm/Support/Errc.h>
#include <llvm/Support/Error.h>
#include <llvm/Support/raw_ostream.h>
#include <vector>
#include <BaseTransferFunction.h>
#include <KillGen.h>
#include <MeetOpInterface.h>
#include <available-support.h>
```

Include dependency graph for dataflow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class llvm::BBInOutBits

  Holds the bitsets for each basic block's IN and OUT. Owner of the memory.
- class llvm::DataflowFramework< D >

  Primary Dataflow Framework template class. Performs the generalized steps of initializing the IN and OUT, calling the Gen and Kill functions and then passing the results to the Transfer Function. Result of which gets set/cleared in the IN/OUT of the correct BB.

Namespaces

- llvm

Enumerations

- enum FlowDirection { FORWARD, BACKWARD }
- enum BoundaryCondition { EMPTY, UNIVERSAL }

7.7.1 Enumeration Type Documentation

7.7.1.1 BoundaryCondition

`enum BoundaryCondition`

Enumerator

EMPTY	
UNIVERSAL	

7.7.1.2 FlowDirection

`enum FlowDirection`

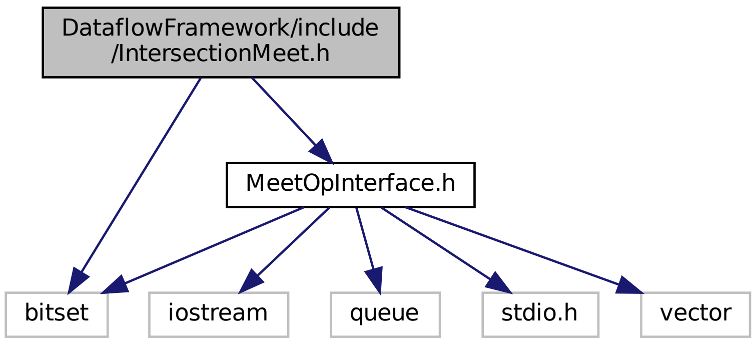
Enumerator

FORWARD	
BACKWARD	

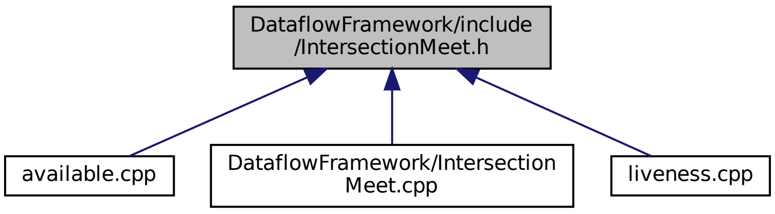
7.8 DataflowFramework/include/IntersectionMeet.h File Reference

```
#include <MeetOpInterface.h>
#include <bitset>
```

Include dependency graph for IntersectionMeet.h:



This graph shows which files directly or indirectly include this file:



Classes

- class IntersectionMeet

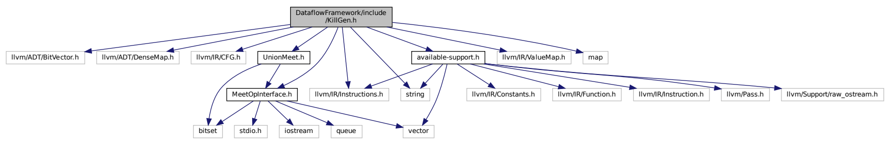
    Intersection meet, implements the meet operation, and is able to set and get the top element.

7.9 DataflowFramework/include/KillGen.h File Reference

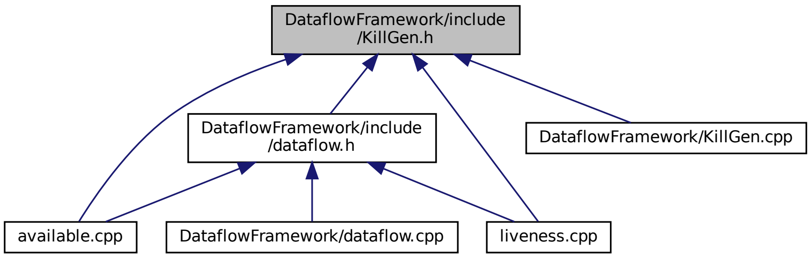
```
#include "llvm/ADT/BitVector.h"
#include "llvm/ADT/DenseMap.h"
#include "llvm/IR/CFG.h"
#include "llvm/IR/Instructions.h"
#include "llvm/IR/ValueMap.h"
#include <MeetOpInterface.h>
#include <UnionMeet.h>
#include <available-support.h>
#include <map>
```

```
#include <string>
```

Include dependency graph for KillGen.h:



This graph shows which files directly or indirectly include this file:


DataflowFramework/include
/KillGen.h
DataflowFramework/include
/dataflow.h
DataflowFramework/KillGen.cpp
available.cpp
DataflowFramework/dataflow.cpp
liveness.cpp

Classes

- class llvm::KillGen< D >

  Interface class for the Kill and Gen functionality. Any new data flow framework must implement their own killEval and genEval functions as they each operate on their domain in their own ways.

Namespaces

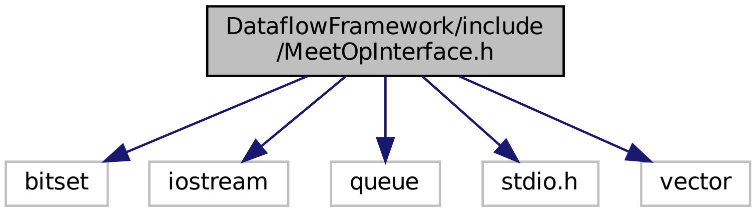
- llvm

7.10 DataflowFramework/include/MeetOpInterface.h File Reference

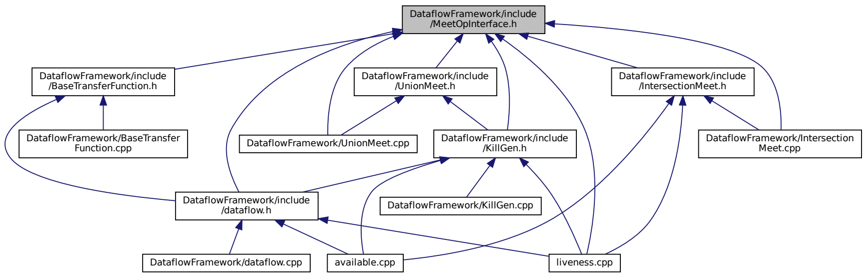
```
#include <bitset>
#include <iostream>
#include <queue>
#include <stdio.h>
```

```
#include <vector>
```

Include dependency graph for MeetOpInterface.h:



This graph shows which files directly or indirectly include this file:


DataflowFramework/include
/MeetOpInterface.h
DataflowFramework/include
/BaseTransferFunction.h
DataflowFramework/include
/UnionMeet.h
DataflowFramework/include
/IntersectionMeet.h
DataflowFramework/BaseTransfer
Function.cpp
DataflowFramework/UnionMeet.cpp
DataflowFramework/include
/KillGen.h
DataflowFramework/Intersection
Meet.cpp
DataflowFramework/include
/dataflow.h
DataflowFramework/KillGen.cpp
DataflowFramework/dataflow.cpp
available.cpp
liveness.cpp

Classes

- class IMeetOp

  MeetOperator pure virtual class. Any new meet operator to be added must inherit this class and implement their own meet function and getters/setters for the top element.

Macros

- #define MAX_BITS_SIZE 4096
- #define MAX_PRINT_SIZE 32

Enumerations

- enum BitsVal { ZEROS, ONES }

7.10.1 Macro Definition Documentation

7.10.1.1 MAX_BITS_SIZE

```
#define MAX_BITS_SIZE 4096
```

7.10.1.2 MAX_PRINT_SIZE

```
#define MAX_PRINT_SIZE 32
```

7.10.2 Enumeration Type Documentation

7.10.2.1 BitsVal

```
enum BitsVal
```

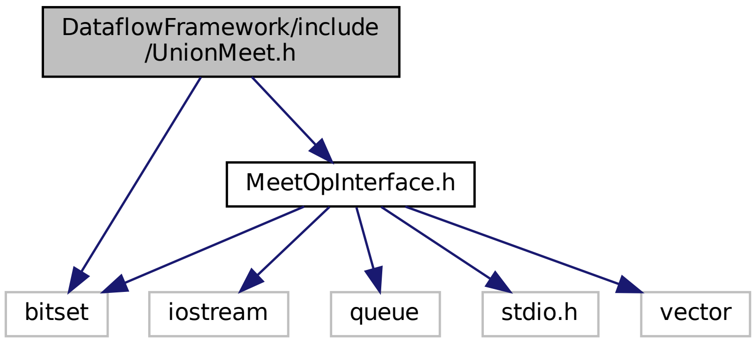
Enumerator

ZEROS	
ONES	

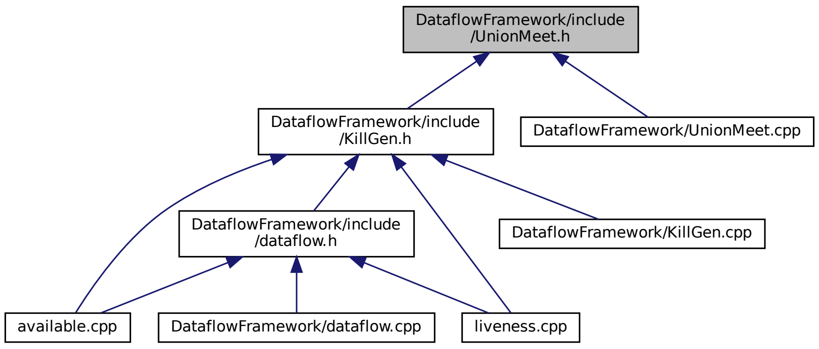
7.11 DataflowFramework/include/UnionMeet.h File Reference

```
#include <MeetOpInterface.h>
#include <bitset>
```

Include dependency graph for UnionMeet.h:



This graph shows which files directly or indirectly include this file:


DataflowFramework/include
/UnionMeet.h
DataflowFramework/include
/KillGen.h
DataflowFramework/UnionMeet.cpp
DataflowFramework/include
/dataflow.h
DataflowFramework/KillGen.cpp
available.cpp
DataflowFramework/dataflow.cpp
liveness.cpp

Classes

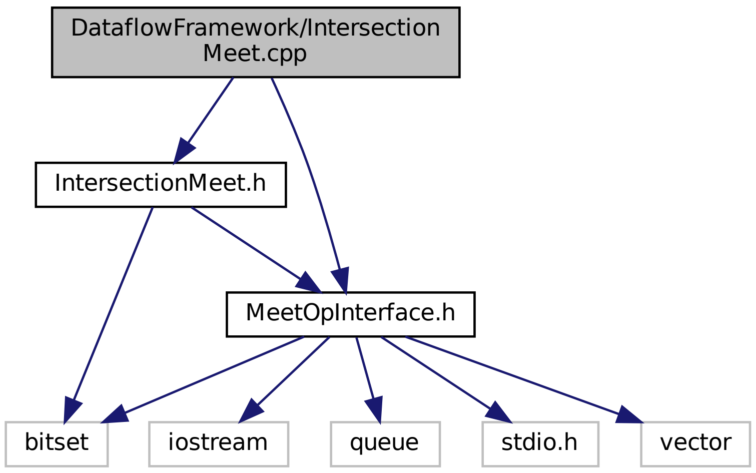
- class UnionMeet

    Union meet, implements the meet operation, and is able to set and get the top element.

7.12 DataflowFramework/IntersectionMeet.cpp File Reference

```
#include <IntersectionMeet.h>
#include <MeetOpInterface.h>
```

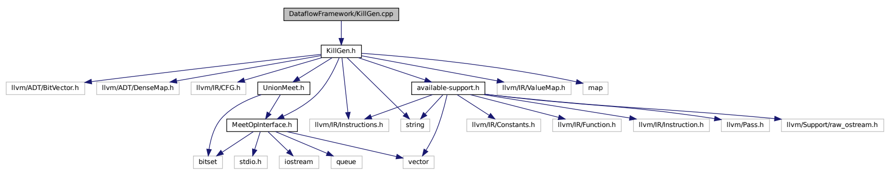
Include dependency graph for IntersectionMeet.cpp:



7.13 DataflowFramework/KillGen.cpp File Reference

```
#include <KillGen.h>
```

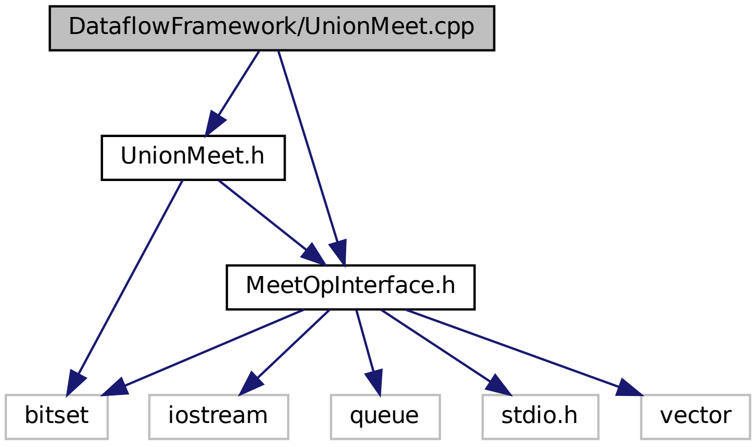
Include dependency graph for KillGen.cpp:



7.14 DataflowFramework/UnionMeet.cpp File Reference

```
#include <MeetOpInterface.h>
#include <UnionMeet.h>
```

Include dependency graph for UnionMeet.cpp:

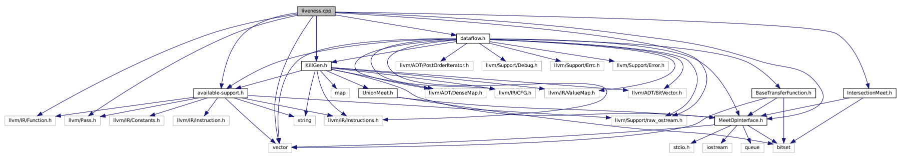


7.15 liveness.cpp File Reference

```
#include "KillGen.h"
#include "MeetOpInterface.h"
#include "dataflow.h"
#include "llvm/IR/Function.h"
#include "llvm/Pass.h"
#include "llvm/Support/raw_ostream.h"
#include <IntersectionMeet.h>
#include <available-support.h>
```

```
#include <vector>
```

Include dependency graph for liveness.cpp:



Classes

- class anonymous_namespace{liveness.cpp}::KillGenLive

  KillGenLive is a subclass of KillGen class, which is a template class. Main function of this class is to provide a killEval and genEval function to take in the input bit set which is the result of the meet operator, the current Basic Block, and the domainset, which is a vector of objects we want to perform the analysis on (variables/Value type in thie case) and returns the resultant kill set or gen set bits respectively.
- class anonymous_namespace{liveness.cpp}::Liveness

Namespaces

- anonymous_namespace{liveness.cpp}

Functions

- RegisterPass< Liveness > anonymous_namespace{liveness.cpp}::X ("liveness", "ECE 5984 Liveness")

7.16 tests/test.c File Reference

Functions

- int main ()

7.16.1 Function Documentation

7.16.1.1 main()

```
int main ( )
```