Assignment 2 Programming Part

Names: Ashwin Krishnakumar, SengMing Yeoh

PID: kashwin , sengming

Directory Structure:

Following is the framework for our assignment 2.

```
/
├── available.cpp
├── available-support.cpp
├── available-support.h
├── compile_commands.json
├── DataflowFramework/
|   ├── BaseTransferFunction.cpp
|   ├── dataflow.cpp
|   ├── include/
|   |   ├── BaseTransferFunction.h
|   |   ├── dataflow.h
|   |   ├── IntersectionMeet.h
|   |   ├── KillGen.h
|   |   ├── MeetOpInterface.h
|   |   └── UnionMeet.h
|   ├── IntersectionMeet.cpp
|   ├── KillGen.cpp
|   └── UnionMeet.cpp
├── liveness.cpp
├── Makefile
```

The “DataflowFramework” directory contains the generalized framework for all data flow analysis. This can be adopted to implement other data flow analysis passes.

Compilation:

To compile the entire project, execute “make” in root directory. This will build both the passes.

Testing:

To execute the passes in default test files given execute “make run_available” or “make run_live”.

For executing the passes on custom files, execute the following command:

1) opt -load ./available.so -available *.bc -o /dev/zero

2) opt -load ./liveness.so -liveness *.bc -o /dev/zero

Design of the framework:

The following were considered as common entities of data flow analysis. We have provided object-oriented interfaces and initialization for assigning the characteristics for the passes.

The constructor for our data flow framework is below. This initializes all the necessary inputs for the framework to perform the analysis.

Code:

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

- Domain, including the semi-lattice

  std::vector<D> &domainset is provided during the construction of the class to the framework. This is used for analyzing corresponding data flow entities. (Expressions, Values)

- Direction (forwards or backwards)

  FlowDirection direction, defines the direction of the flow of analysis. FORWARD iterates the graph in inverse post order. And BACKWARD iterated the graph in post order fashion.

• Transfer function

BaseTransferFunction &transfer, defines the transfer function of the analysis. The KillGen<D> &KillGenImp template gives us interface to define the context of kill and gen set for the particular analysis in the framework.

• Meet Operation

The IMeetOp &meetOp operator defines the meet functionality of the analysis. This interface should be extended to a specific meet operation by inheriting the IMeetOp class.

• Boundary condition (entry or exit)

The BoundaryCondition boundary variable defines which basic block must be set or rest depending upon the nature of meet operator. (If the meet operator is intersection,we have to initialize the INs of the BB to set , if the flow is FORWARD and set the OUTs of BBs to set if flow BACKWARDS)

• Initial interior points (in or out)

We derive it internally in the dataflow from the direction and the meet operator nature.