ECE CS 5544 - Assignment 2 - Part 1

Students

- Swati Lodha - swatil
- Abhijit Tripathy - abhijittripathy

Implementation

The project consists of a generic dataflow framework and implementations for Available Expressions, Liveness Analysis and Reaching Definitions passes. The framework uses `BitVector` to represent the `domain`, `genSet` and `killSet`. The dataflow pass can be configured using it's constructor. It also exposes the `executeDataFlowPass` API to run the pass. The constructor details for dataFlow is as follows:

```
dataFlow(int ds, enum meetOperator m, enum passDirection p, BitVector b,
BitVector i)
```

where,

- ds : domain size
- m : Meet Operator (`UNION` | `INTERSECTION`)
- p : Pass Direction (`FORWARD` | `BACKWARD`)
- b : `BitVector` representing the boundary conditions
- i : `BitVector` representing the initial conditions

The `dataFlow.cpp` class implements two data structures to represent the DFA information for each BasicBlock.

```
struct basicBlockDeps {
  BasicBlock *blockRef;
  BitVector genSet;
  BitVector killSet;
};

struct basicBlockProps {
  enum blockType bType;
  BasicBlock *block;
  BitVector bbInput;
  BitVector bbOutput;
  BitVector genSet;
  BitVector killSet;
  std::vector<BasicBlock *> predBlocks;
  std::vector<BasicBlock *> succBlocks;
};
```

It also exposes the `executeDataFlowPass` API that takes the following params :

- `Function &F` : Reference to the Function for which the pass is being run.
- `std::map<BasicBlock*, basicBlockDeps> bbMap` : Mapping of Basic Blocks to their `genSet` and `killSet`.

The framework also internally implements the `applyMeet` and `applyTransferFn` sub-routines that calculates `IN[BB]` and `OUT[BB]` for each BasicBlock.

The project README contains information on building and running the three DFA passes.