Final Project: Data Flow Analysis

S. Patel, J. Collard, M. Barney

May 3, 2013

1 Introduction

This report contains our implementation of a scanner and parser for a basic programming language, and our data flow graph generation tools.

It is divided up into several sections, roughly corresponding to the problems given in the specification, each a Haskell module.

2 Abstract Syntax

In this module we define the abstract syntax (AST) for statements written in a simple imperative language.

```
module AST where
data AOP =
  Plus |
  Times |
  Minus deriving (Eq, Show)
data BOP =
  And |
  Or deriving (Eq, Show)
data REL =
  Equal |
  Less |
  Leq |
  Greater |
  Geq deriving (Eq, Show)
data Arith =
  Var String |
```

```
  Number Int |
  BinOp AOP Arith Arith deriving (Eq, Show)
data Boolean =
  T |
  F |
  Not Boolean |
  BoolOp BOP Arith Arith |
  RelOp REL Arith Arith deriving (Eq, Show) kjsdfjlkjkldsjfsdkkldsj
data Statement =
  Assign String Arith |
  Skip |
  Seq Statement Statement |
  If Boolean Statement Statement |
  While Boolean Statement deriving (Eq, Show)
```

3 Main module

The main module puts everything together.

```
module Main where
import System.Environment
import AST
main = do
  [file] ← getArgs
  contents ← readFile file
  print contents
```