SE 421: Assignment #4

Due on October 1, 2018 at 12:00 PM (noon)

Instructor: Ben Holland

https://github.com/SE421/assignment4

Student Name:

Problem 1

(12 points)

A. (*1.5 points*) Given the following program, how many live definitions could reach the use of "`a`" in the assignment to "`b`"? Explain.

```
1   int a = 0;
2   if(C1){
3       a = 1;
4       if(C2){
5           a = 2;
6           if(C3){
7               a = 3;
8           } else {
9               a = 4;
10          }
11      } else {
12          a = 5;
13      }
14  } else {
15      a = 6;
16  }
17  int b = a;
18
```

B. (*1.5 points*) Given the following program, how many live definitions could reach the use of "`a`" in the assignment to "`b`"? Explain.

```
1   int a = 0;
2   if(C1){
3       a = 1;
4   } else {
5       a = 2;
6   }
7   if(C2){
8       a = 3;
9   } else {
10      a = 4;
11  }
12  if(C3){
13      a = 5;
14  } else {
15      a = 6;
16  }
17  int b = a;
18
```

C. (*1.5 points*) Given the following program, how many uses of "`a`" (in assignments to "`b`") correspond to the definition of "`a`"? Explain.

```
int a = 0;
int b;
if(C1){
    b = a;
    if(C2){
        b = a;
        if(C3){
            b = a;
        } else {
            b = a;
        }
    } else {
        b = a;
    }
} else {
    b = a;
}

```

D. (*1.5 points*) Given the following program, how many uses of "`a`" (in assignments to "`b`") correspond to the definition of "`a`"? Explain.

```
int a = 0;
int b;
if(C1){
    b = a;
} else {
    b = a;
}
if(C2){
    b = a;
} else {
    b = a;
}
if(C3){
    b = a;
} else {
    b = a;
}

```

E. (*1 points*) TRUE / FALSE. There cannot be more reaching (live) definitions for a variable than there can be uses for a variable. Explain.

F. (*3 points*) List the different programming language features that allow a function `foo` to pass data to or from another function `bar`. For each language feature, provide an example and indicate the underlying memory architecture (stack or heap) that enables the transfer of information.

G. (*2 points*) Can data flow be obscured through control flow? Explain. Likewise can control flow be obscured through data flow? Explain.

Problem 2

(8 points)

A. (*6 points*) Points-to Analysis

In this problem, you are required to modify the provided skeleton code (`AndersenPointsTo.java`) to compute a pointer analysis. Your implementation should adhere to the following requirements:

- Your implementation must support analysis of Java programs, analysis of C pointers is not required for this assignment.
- Your implementation must save results as Atlas tags, where each tag is unique to a particular heap allocation ("new" instantiation) and is applied to all variables (i.e. data flow nodes) that may alias each other.

Commit your changes to the code on your GitHub assignment repository and briefly explain your modifications. Remember to provide a direct link to your source code on GitHub in your report!

The provided code is an Eclipse plugin that depends on Atlas (`http://www.ensoftcorp.com/atlas`). You will be using the Atlas SDK (`https://www.ensoftcorp.com/atlas/sdk`) and its XCSG graph schema (`http://ensoftatlas.com/wiki/Extensible_Common_Software_Graph`) to extend the functionality of the provided plugin project. You may wish to use tools or utilities from your previous assignment.

B. (*2 points*) Experimentation

Using your Points-to implementation run your plugin project as an Eclipse Plugin application. Eclipse will launch a new runtime copy of Eclipse. The skeleton project includes a source file called `PointsToAnalysisCodemapStage.java`, which configures your Points-to analysis to run automatically after Atlas produces a codemap for a project. The "Element Detail View" provided by Atlas is useful for inspecting the alias tags applied by your analysis. You may also find the Atlas Data Flow "Smart View" useful for understanding the data flow graphs that are provided by Atlas. Finally, the skeleton project includes a custom Atlas "Smart View" defined in `PointsToAliasesSmartView.java` that allows you to select a data flow node and view the corresponding instantiation node with the data flow path to the selected node.

The Andersen-style points-to analysis is a flow-insensitive algorithm. Using these tools determine if the results that your algorithm produced were flow sensitive or flow insensitive. Explain your findings and provide supporting evidence. You may find it useful to review materials on Static Single Assignment form and experiment with the Atlas data flow graph to explain your results.

C. (*0 points*) Unit Testing

A JUnit test framework with some example unit tests have been provided. To run the unit tests, right click on the `AllTests.java` file in the `com.se421.pointsto.regression` project and select `Run As` → `JUnit Plug-in Test`. To fully test your code you should extend these unit tests with your own test cases! Note that you are allowed to share and discuss test cases with other students as long as you cite your sources and explain the included content in your own words!

Problem 3

Extra Credit (15+ points)

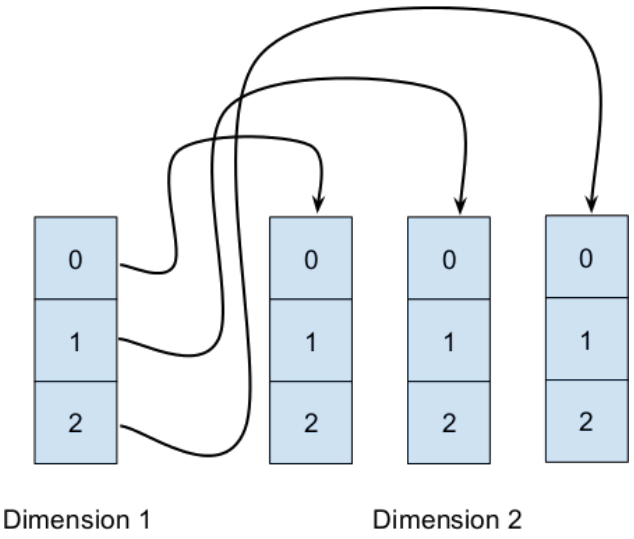
A. (*5 points*) Type Inference

Atlas conservatively adds an `XCSG.TypeOf` edges (see `https://ensoftatlas.com/wiki/XCSG:TypeOf`) from variables to the *declared* type of an object. For example in the statement `Animal a = new Dog();`, Atlas would add a type of edge from "`a`" to the type "`Animal`" because "`a`" was declared to be a type of Animal even though "`a`" is actually instantiated with the type of "`Dog`". Since a points-to analysis tracks the instantiations that may reach each variable it would be a simple task to create a new edge, which we will call an `INFERRED_TYPE_OF` edge from the variable "`a`" to the type "`Dog`". The skeleton code provided to you will add these edges for you if you complete the `getInstantiatedType` method. Note that you are free to change this method or anything else in the `AndersenPointsTo.java` source file to complete your objective.

B. (*10 points*) Array Support

For this task you will be attempting to adds support for tracking objects through arrays. You should propose a strategy and provide an implementation of your strategy. A variable amount of points will be awarded based on the level of precision that you add while balancing any tradeoffs in cost that you incur as overhead for your added precision.

Note that multi-dimensional arrays in Java are a source level artifact only. At the bytecode level multi-dimensional arrays are implemented as single dimensional arrays with object references to other arrays as indicated in the included figure.



C. (*? points*) Email the instructor and pitch an enhancement to implement. Possible themes are exploring the use binary decision diagrams for scalability, using strongly connected components to improve performance, adding support of resolving inter-procedural data flows, adding context sensitivity, implementing a particular client analysis of points-to analysis, etc.