Piacente Cristian 866020

Assignment A6 – Generalized Symbolic execution

Software Quality, Academic Year 2023-2024, University of Milan – Bicocca

Symbolically execute method enforcePosNeg, and determine whether there can be any violation of the assertion in the program. Motivate your answer by showing the path conditions that you generated during the analysis, also stating for each of them whether it is satisfiable or not.

If the assertion can be violated, provide concrete inputs that lead the system to reach it, and show that these input are a solution the path condition computed with the symbolic execution.

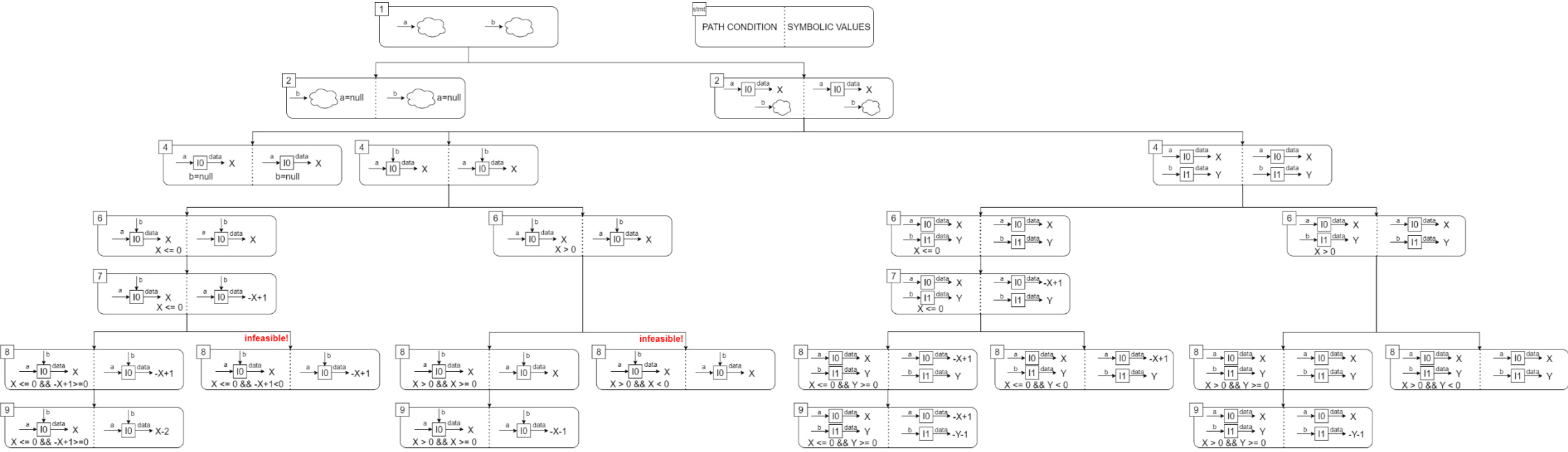
Say also how many feasible paths there are in the considered method, motivating your answer based on your analysis.

```
public class Item {
   private int data;

   public Item(int data) {
     this.data = data;
   }

1  public static void enforcePosNeg(Item a, Item b) {
2    if (a == null) {
3      return;
     }
4    if (b == null) {
5      return;
     }
6    if (a.data <= 0) { // enforce positive value
7      a.data = -a.data + 1;
     }
8    if (b.data >= 0) { // enforce negative value
9      b.data = -b.data - 1;
     }
10   assert(a.data > b.data);
   }
}
```

Piacente Cristian 866020

Assignment A6 – Generalized Symbolic execution

Software Quality, Academic Year 2023-2024, University of Milan – Bicocca



To understand the solution properly, the following notation has been used:

- the cloud refers to a reference which has not been used, so it could be a null reference or an instance of Item
  - according to my notation, the int property “data” is never represented as a cloud because an int value can’t be null in Java
- I0 and I1 are the symbolic values for the parameters “a” and “b”, which are objects of type Item
- X and Y are the symbolic values for the int properties “a.data” and “b.data”.

Let’s delve into the analysis of each execution path (found by looking at the leaves of the tree) and determine if there can be any violation of the assertion.

- **Path 1-2-3**, path condition **a==null**
  This path is **feasible** (the path condition is **satisfiable**), but it does not reach the assert statement, since the execution stops at the return statement at line 3, so **the assertion can’t be violated**.

- **Path 1-2-4-5**, path condition **a!=null && b==null**
  This path is **feasible** (the path condition is **satisfiable**), but it does not reach the assert statement, since the execution stops at the return statement at line 5, so **the assertion can’t be violated**.

Piacente Cristian 866020

Assignment A6 – Generalized Symbolic execution

Software Quality, Academic Year 2023-2024, University of Milan – Bicocca

- **Path 1-2-4-6-7-8-9-10**, path condition **a!=null && b aliases a && a.data <= 0 && -a.data+1>=0**
  This path is **feasible** (the path condition is **satisfiable**), we can check if the assertion can be violated or not by checking if the following condition is satisfiable:
  a!=null && b aliases a && a.data <= 0 && -a.data+1>=0 && a.data-2 <= a.data-2
  The assertion can be violated; note that since b aliases a, we can swap a.data with b.data in our condition without making any change.
  We can provide a **concrete test case** to violate the assertion, using 0 as the value of a.data (this condition is true: 0 <= 0 && 1 >= 0):
  Item a = new Item(0);
  Item b = a;
  Item.enforcePosNeg(a, b);

- **Path 1-2-4-6-7-8-10**, path condition **a!=null && b aliases a && a.data <= 0 && -a.data+1<0**
  This path is **infeasible** (the path condition is **not satisfiable**, since a.data can't be both less or equal to 0 and greater than 1): **the assertion can't be violated**.

- **Path 1-2-4-6-8-9-10**, path condition **a!=null && b aliases a && a.data > 0 && a.data >= 0**
  This path is **feasible** (the path condition is **satisfiable**), we can check if the assertion can be violated or not by checking if the following condition is satisfiable:
  a!=null && b aliases a && a.data > 0 && a.data >= 0 && -a.data-1 <= -a.data-1
  The assertion can be violated; note that since b aliases a, we can swap a.data with b.data in our condition without making any change.
  We can provide a **concrete test case** to violate the assertion, using 1 as the value of a.data (this condition is true: 1 > 0 && 1 >= 0):
  Item a = new Item(1);
  Item b = a;
  Item.enforcePosNeg(a, b);

- **Path 1-2-4-6-8-10**, path condition **a!=null && b aliases a && a.data > 0 && a.data < 0**
  This path is **infeasible** (the path condition is **not satisfiable**, since a.data can't be both greater than 0 and less than 0): **the assertion can't be violated**.

- **Path 1-2-4-6-7-8-9-10**, path condition **a!=null && b!=null && !(b aliases a) && a.data <= 0 && b.data >= 0**
  This path is **feasible** (the path condition is **satisfiable**), we can check if the assertion can be violated or not by checking if the following condition is satisfiable:
  a!=null && b!=null && !(b aliases a) && a.data <= 0 && b.data >= 0 && -a.data+1 <= -b.data-1
  The assertion can't be violated: a.data-2 >= b.data has no solutions, with the constraints of a.data being non-positive and b.data being non-negative.

- **Path 1-2-4-6-7-8-10**, path condition **a!=null && b!=null && !(b aliases a) && a.data <= 0 && b.data < 0**
  This path is **feasible** (the path condition is **satisfiable**), we can check if the assertion can be violated or not by checking if the following condition is satisfiable:
  a!=null && b!=null && !(b aliases a) && a.data <= 0 && b.data < 0 && -a.data+1 <= b.data
  The assertion can't be violated: -a.data+1 <= b.data has no solutions, with the constraints of a.data being non-positive and b.data being negative.

- **Path 1-2-4-6-8-9-10**, path condition **a!=null && b!=null && !(b aliases a) && a.data > 0 && b.data >= 0**
  This path is **feasible** (the path condition is **satisfiable**), we can check if the assertion can be violated or not by checking if the following condition is satisfiable:
  a!=null && b!=null && !(b aliases a) && a.data > 0 && b.data >= 0 && a.data <= -b.data-1

Piacente Cristian 866020

Assignment A6 – Generalized Symbolic execution

Software Quality, Academic Year 2023-2024, University of Milan – Bicocca

The assertion can't be violated: a.data <= -b.data-1 has no solutions, with the constraints of a.data being positive and b.data being non-negative.

- **Path 1-2-4-6-8-10**, path condition **a!=null && b!=null && !(b aliases a) && a.data > 0 && b.data < 0**
  This path is **feasible** (the path condition is **satisfiable**), we can check if the assertion can be violated or not by checking if the following condition is satisfiable:
  a!=null && b!=null && !(b aliases a) && a.data > 0 && b.data < 0 && a.data <= b.data
  The assertion can't be violated: a.data <= b.data has no solutions, with the constraints of a.data being positive and b.data being negative.

In conclusion, we have analyzed each execution path; for those which have a satisfiable path condition and which reach the assert statement, we have checked if the assertion can be violated by checking if path condition && a.data[S_i] <= b.data[S_i] is satisfiable (where S_i is the symbolic state of the path i, so we consider the values of a.data and b.data from the symbolic state). We have provided a test case for each execution path which can violate the assertion.

In total, we have analyzed **10 execution paths** (we have generated 10 path conditions which characterize the paths): **8 of them are feasible**, and **the assertion can be violated in 2 of the 8 feasible paths**, so we have generated 2 concrete test cases for violating the assertion.