Program Analysis, University of Stuttgart, Winter 2020/21

Exercise 3: Information Flow and Slicing

Deadline for uploading solutions via Ilias:
January 29, 2021, 11:59pm Stuttgart time

Task 1 Information Flow Analysis [20 points]

This task is about dynamic information flow analysis. Consider the following JavaScript code to analyze:

```
var age = getAge();
var flag = getFlag();
var count = 0;
var mult = 2;
if (age === 20){
  count++;
} else {
  if (age === 25){
    flag = 2;
  } else {
    if (age === 30) {
      flag = 3;
    }
  }
}

if (flag === 2){
  flag = flag * mult;
  print_public(flag);
} else {
  print_public(mult);
}
```

There are three security classes: *secret*, *confidential*, and *public*, which are ordered into a lattice such that:

$$secret > confidential > public$$

By default, all values are labeled as *public*. Values returned by `getAge()` are labeled as *secret* and values returned by `getFlag()` are labeled as *confidential*. The function `print_public()` is an untrusted sink, which should only be reached by *public* information. Note that passing an argument to function should be handled like an assignment to the formal parameter of the function.

Subtask 1.1 Execution 1 [10 points]

Consider a dynamic information flow analysis that considers both explicit and implicit flows. Suppose an execution where `getAge()` returns 20 and `getFlag()` returns 3.

- What are the security labels of variables and expressions during the execution? Use the following template to provide your answer.

Line	Variable or expression	Security label of variable or expression (after executing the line)
1	`age`	
2	`flag`	
3	`count`	
4	`mult`	
5	`age === 20`	
6	`count`	
17	`flag === 2`	

- Does the execution violate the information flow policy? Explain your answer.

Subtask 1.2 Execution 2 [10 points]

Consider a dynamic information flow analysis that considers both explicit and implicit flows. Suppose an execution where `getAge()` returns 25 and `getFlag()` returns 1.

- What are the security labels of variables and expressions during the execution? Use the following template to provide your answer.

Line	Variable or expression	Security label of variable or expression (after executing the line)
1	`age`	
2	`flag`	
3	`count`	
4	`mult`	
5	`age === 20`	
8	`age === 25`	
9	`flag`	
17	`flag === 2`	
18	`flag`	

- Does the execution violate the information flow policy? Explain your answer.

Task 2 Universally Bounded Lattice [20 points]

In this task we will analyze the characteristics and properties of universally bounded lattices.

Subtask 2.1 Recognize a Universally Bounded Lattice [10 points]

Figure 1 shows structures that could represent universally bounded lattices.

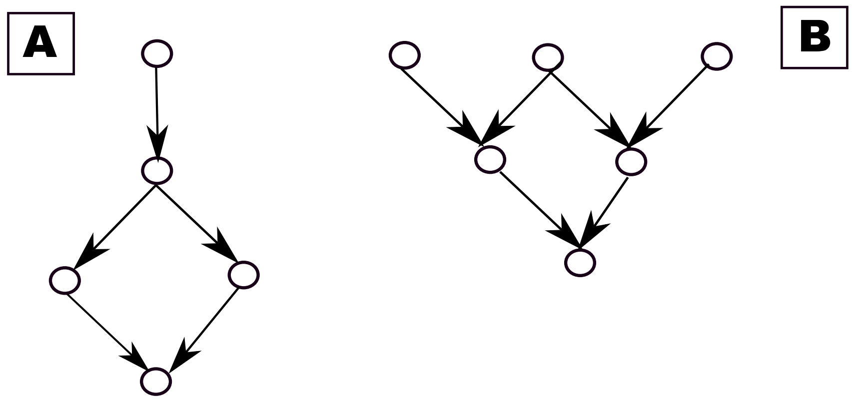
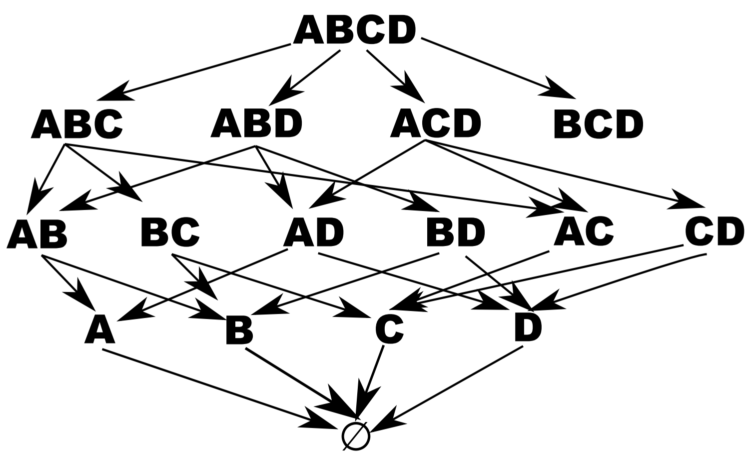


Figure 1: Two universally bounded lattice candidates.

- Are the two candidates universally bounded lattices? Explain why (not).

Subtask 2.2 Characteristics [10 points]

Consider the following structure that represents a universally bounded lattice:



Answer the following questions:

- Give the set S of security classes.

  $S =$

- What is the lower bound $\perp$?

  $\perp =$

- What is the upper bound $\top$?

  $\top =$

- Let $\bigoplus$ be the least upper bound operator. What is the result of the following operations?

  $B \bigoplus BC =$

  $ABCD \bigoplus CD =$

  $ABC \bigoplus \emptyset =$

- Let $\bigotimes$ be the greatest lower bound operator. What is the result of the following operations?

  $D \bigotimes DC =$

  $ACD \bigotimes AD =$

  $ABCD \bigotimes B =$

Task 3 Static Slicing as Proposed by Weiser [30 points]

Consider the following JavaScript program:

```
1  var year = getYear();
2  var flag = getFlag();
3  var count = 0;
4  var mult = 3;
5  if (year === 2020) {
6    count++;
7    mult = mult * count;
8  } else {
9    if (year === 2021) {
10     flag = 2 + flag;
11   }
12 }
13 var sliceHere = count;
```

Compute the static backward slice for variable `sliceHere` at line 13. Use the slicing approach of Weiser (IEEE TSE, 1984) and its formulation as a graph reachability problem, as it has been introduced in the lecture. To describe your solution, follow the steps outlined below.

Subtask 3.1 Data Flow Dependences [10 points]

Provide the data flow dependences between statements in the program. Use the following table to summarize the dependences. Each table cell represents a pair of statements. Mark all pairs of statements that have a definition-use relationship.

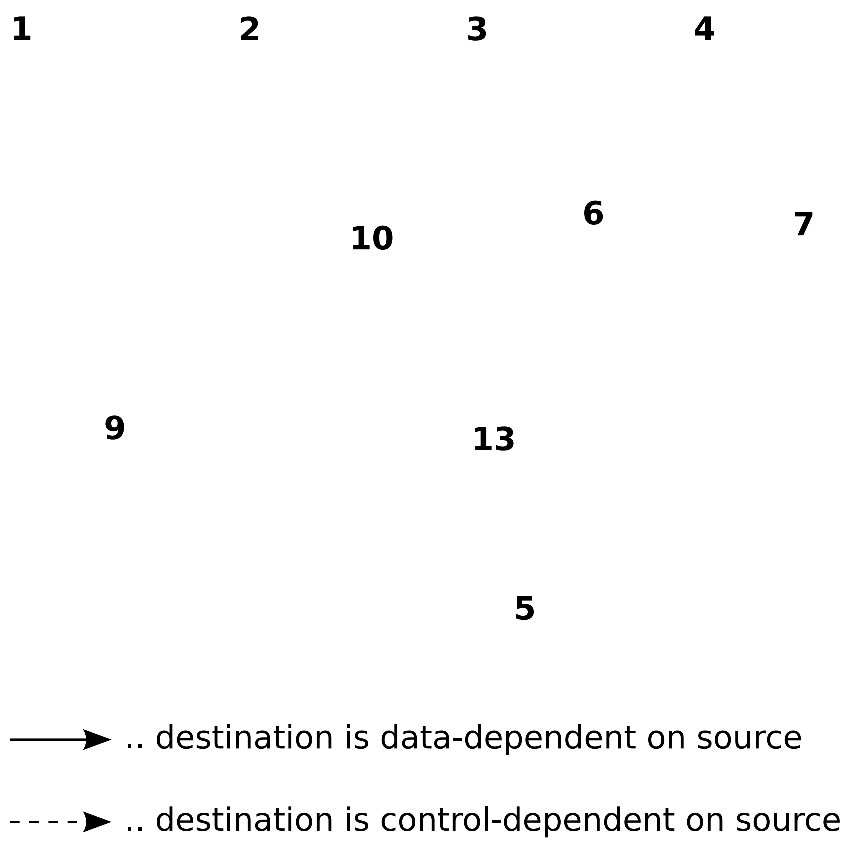
Def	**Use**									
	1	2	3	4	5	6	7	9	10	13
1										
2										
3										
4										
5										
6										
7										
9										
10										
13										

Subtask 3.2 Control Flow Dependences [10 points]

Provide the control flow dependences between statements in the program. Describe your solutions as a sequence of "Statement .. is control-flow dependent on statement .." sentences.

Subtask 3.3 Program Dependence Graph [5 points]

Summarize the data flow dependences and the control flow dependences into a program dependence graph. Use the following template to draw your solution.



Subtask 3.4 Slice [5 points]

What is the slice for the slicing criterion (variable sliceHere at line 13)? Write down the source code of the sliced program as a syntactically correct program.

Task 4 Dynamic Slicing (Revised Approach) [30 points]

Consider the following JavaScript program:

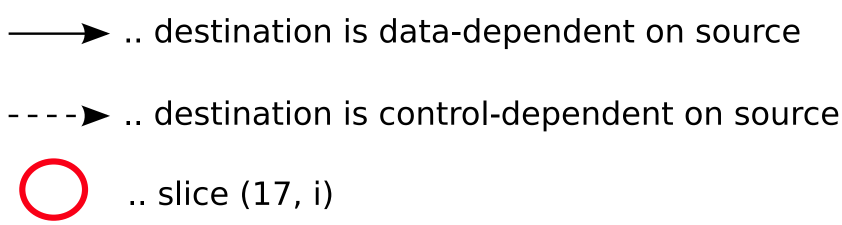
```
1  var year = getInput();
2  var actual_year = 2021;
3  var count = 0;
4  var flag = false
5  var diff = actual_year - year
6  var i = 0;
7  while (i < diff) {
8    i++;
9    year++;
10   if (year > 2000) {
11     count++;
12   }
13 }
14 if (count > 10) {
15   flag = true
16 }
17 console.log(i)
```

Subtask 4.1 Execution History [5 points]

Give the execution history with *getInput*() = 2018. You can use the line numbers to refer to statements. Do not include the lines with the closing curly brackets (}) into the history.

Subtask 4.2 Dynamic Dependence Graph **[20 points]**

Suppose we want to compute the dynamic backward slice with the last statement (*console.log(i)*) as the slicing criterion. Use $getInput() = 2019$ (note that this input is different from above). Provide the dynamic dependence graph, using the revised approach presented in the lecture.



Subtask 4.3 Slice [5 points]

Write the sliced program.