

**Report on “Static Program Analysis by using LiSA”**

**Prepared by: Anees Baqir [956610]**

**Course\_Name: ACADIA-2 [PHD156]**

**Instructor: Prof. Pietro Ferrara**

## 1. Introduction:

Modern programming languages make extensive use of strings. Their uses range from providing a user with an output to the development of programs that are implemented through reflection. For example, strings in PHP can be used to communicate between programs, while strings in Java are commonly used as SQL queries or to access information about classes through reflection. For all these reasons, interest in methods for automatically analyzing and detecting bugs in strings is growing all the time. The state-of-the-art in this area, on the other hand, is still limited [1]. This report is based on the abstract interpretation of string analysis using Library for Static Analysis (LiSA) [2].

In this report, we will refer to the following two examples for the static analysis as **Program a** and **Program b**.

```
***

class coffee_program {
    getPrices(lowerbound) {
        def query = "SELECT '$' || (RETAIL_PRICE/100) from COFFEES WHERE
";
        if (lowerbound != null) {
            query = strcat(query, strcat(strcat("WHOLESALE_PRICE > ",
lowerbound), " AND "));
        }
        query = strcat(query, strcat(strcat("TYPE IN (", "SELECT
COF_NAME, SUP_ID, PRICE FROM COFF_TYPE WHERE COF_NAME = 'mocca' OR COF_NAME =
'espresso'"), ", ");");
        return query;
    }
}
```

### Program a

```
*****

class product_price {
    getPrices(lowerbound) {
        def query = "SELECT Product_Price.Date, Product_Price.Product,
Product_Price.MinimumSellerName FROM (SELECT MIN(Price) AS min_price,
Product, Date FROM Product_Price GROUP BY Product, Date)
min_price INNER JOIN Product_Price ON min_price.Product =
Product_Price.Product
```

```

        AND min_price.Date = Product_Price.Date";

        if (lowerbound != null) {

            query = strcat(query, strcat(strcat("ACTUAL_PRICE > ",
lowerbound), " AND "));

        }

        query = strcat(query, strcat(strcat("TYPE IN (", "SELECT
Product_Name, Dealer_ID, FROM PRODUCT_TYPE WHERE Product_Name = 'glass' OR
Product_Name = 'leather'"), " ");");

        return query;

    }

}

```

### Program b

The code fragment **Program a** highlights some typical programming mistakes that Java servlet application developers make. The `getPrices` method builds the query string to hold a SQL SELECT statement to return the prices for all perishable goods and runs the query. In the code, `||` is the operator of the concatenation and Form IN clause (...) tests whether the type code Class fits either of the types of objects that could be perished. If `lowerbound` is "595," the query which will be executed is as follows:

```

SELECT '$' || (RETAIL_PRICE/100) from COFFEES
    WHERE WHOLESale_PRICE > 595 AND TYPE IN
        (SELECT COF_NAME, SUP_ID, PRICE FROM COFF_TYPE WHERE COF_NAME =
        'mocca' OR COF_NAME = 'espresso');

```

With the above example, followerbounding runtime errors can arise [3]:

- I. The expression `'$' || (RETAIL_PRICE/100)` concatenates the symbol '\$' with the numeric outcome of expression `RETAIL_PRICE/100`. Although certain database systems tap into a string implicitly, a lot of them fail and make a runtime error.
- II. Considering the expression of `WHOLESale_PRICE > lowerbound`, the type of variable `lowerbound` is string and the type of column `WHOLESale_PRICE` is integer. No problem can occur until the variable `lowerbound` is a string representing a number, but this is risky.

And for this program, we aim to check if the concatenation which applies only if a particular value is not null. Moreover, we aim to check if the resulting SQL query of this code is well formed. The results obtained by applying our analysis are shown in the figures below. And in the following sections, we discuss the methods of abstract domains and semantics applied in the analysis.

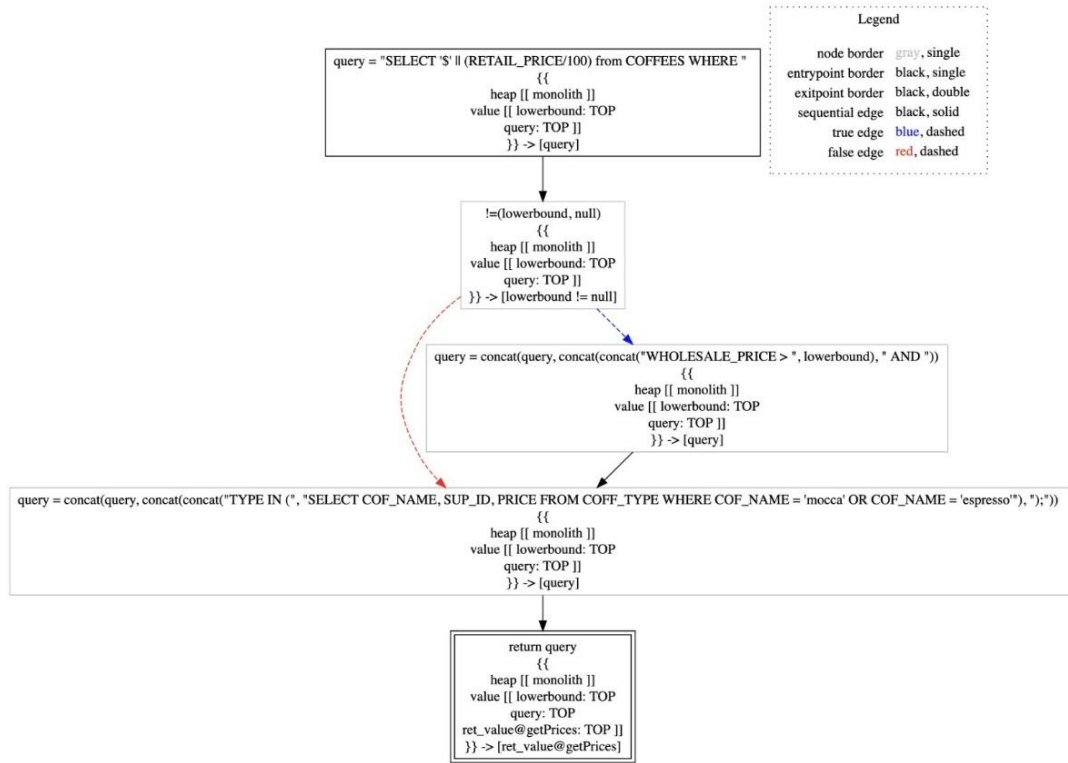


Figure 1: Program a

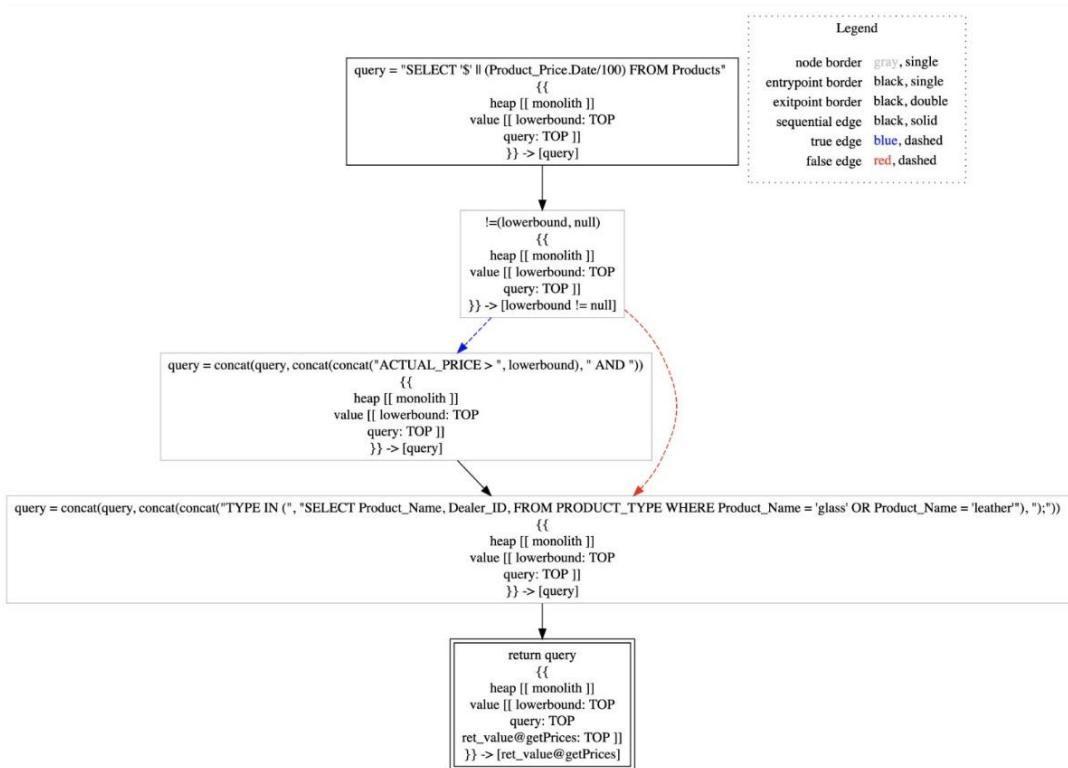


Figure 2 Program b

## 2. Character Inclusion:

We know that when we test a string the characters that are sure or even included are only the characters that exist in the string. The two strings concatenation contains all the characters surely or even used in these two strings. In our example in , we do not know the value of `lowerbound`, after instruction 1 and 3, we calculate the minimum upper bound between the abstract query values. This gives us the abstraction after the `if` expression of the abstract query value. Then the query contains all the characteristics at the end of the given code and could contain a character, as we would have linked a string to the input query (`lowerbound`).


## 3. Prefix and Suffix:

For program a, at the first line the query contains the whole string (`STR1`) as both prefix and suffix. As mentioned in previous section, we do not know the value of `lowerbound`. However, when we concatenate the value of `lowerbound`, of the resulting string we have some information on the prefix and suffix. Therefore, at the end of analyses we find the prefix of the query is string `STR1`. And its suffix is the last query of the program, however, we lose information in the middle.

## 4. Bricks:

This abstract domain works by capturing inclusion and order among the characters of the query by employing simplified regular expressions. As a result, the knowledge collected by this domain may be used to demonstrate more complex properties than the previous domains. A brick represents all the strings that can be formed by combining the specified strings between min and max times. We reflect query with a single brick and a singleton set in line 4. Since the value of '`lowerbound`' is not defined or known previously, it is linked to TOP. At line 6, we concatenate the value of query to '`lowerbound`', '`WHOLESALE_PRICE`' and '`AND`' , resulting in a sequence of four bricks: the first two are made up by a singleton set (containing, respectively, line 4 and '`WHOLESALE_PRICE`', the third is TOP (due to '`lowerbound`', and the fourth is made up by a singleton set (containing '`AND`'). This means that the string associated with the question begins with line 4 '`WHOLESALE_PRICE`', then has an undisclosed section, and finally ends with '`AND`'.

## References

- [1] G. Costantini, P. Ferrara, and A. Cortesi, "Static analysis of string values," in *International Conference on Formal Engineering Methods*, 2011, pp. 505–521.
- [2] "GitHub - UniVE-SSV/lisa:  LiSA, a Library for Static Analysis." [Online]. Available: <https://github.com/UniVE-SSV/lisa>. [Accessed: 30-Apr-2021].

- [3] C. Gould, Z. Su, and P. Devanbu, "Static checking of dynamically generated queries in database applications," in *Proceedings. 26th International Conference on Software Engineering*, 2004, pp. 645–654.