Jtrix

Arick Huang (ah3445) Azim Djuraev (ad3390)
Jacob Shiers (jms2453) Levi Beckett (lb3064)

February 20, 2019

1 Overview

We aim to create a language that can facilitate matrix manipulation. We plan to give the user basic primitive types such as int, string, boolean, char, and float. We also will add matrix and confusion matrix primitives. We will require type verification of the primitives and functions to increase readability and we will type them using our compiler. Along with these primitives, we plan to offer simple operations for integers and floats such as multiplication, division, addition, and subtraction and strings and chars such as concatenation and comparison. Our matrix operations will consist of addition, subtraction, multiplication and transposing. For confusion matrices, we will create standard library functions that will allow the user to calculate confusion matrix statistics. To give the code simplicity and readability for a large audience our syntax will be very similar to Java.

2 Team Roles

Manager: Azim Djuraev
Language Guru: Jacob Shiers
System Architect: Levi Beckett
Tester: Arick Huang

3 Motivation

Our motivation stems from the necessity of matrices in machine learning and the growing need for machine learning in technology and nearly every other sector. Many languages that offer powerful matrix manipulation, such as Haskell, are difficult to learn for people who are new to programming. By using Java-like syntax, our language can serve as a bridge for inexperienced coders looking to utilize matrices in their programming.

4 Language Features

4.1 Data Types

In our program, we aim to add the following primitives similar to what one would see in Java:

- int
- float
- boolean
- char
- string

Furthermore, we would add a new data type: Matrix that should contain only integers or floats.
We also plan to add structs that will contain many variables with unique names so that one can access them easily and funcs that will hold functions.

4.1.1 Example

```
1:  Matrix<2, 1> newMatrix = [1 | 2];
```

4.1.2 Example

```
1:  struct structure = {
2:       Matrix m1 = [];
3:  }
```

4.2 Control Flow

The Jtrix language supports `if, else if, else` statements for conditionals, `for, foreach, while` as loops.

4.2.1 Conditional Statements

One would construct an if statement as shown below:

```
1:  if (boolean expression) {
2:       Statements
3:  }
4:  else if (boolean expression) {
5:       Statements
6:  }
7:  else {
8:       Statements
9:  }
```

4.2.2 Loops

One would contruct a for loop as shown below:

```
1:  for (int i = 0; i < 10; i = i + 1) {
2:       Statements
3:  }
```

One would construct a foreach loop as shown below:

```
1:  foreach (x in matrix) {
2:       Statements
3:  }
```

One would construct a while loop as shown below:

```
1:  while (i < 10) {
2:       Statements
3:  }
```

4.3 Operators

We plan to have the following operations for our language:

Arithmetic Operations	Applies To
Multiplication (a * b)	int, float, Matrix
Division (a / b)	int, float
Modulo (a % b)	int
Addition (a + b)	int, float, Matrix
Subtraction (a - b)	int, float, Matrix

Relational Operations	Applies To
Equality (a == b)	int, boolean, float, Matrix
Inequality (a != b)	int, boolean, float, Matrix
Greater than (a > b)	int, float
Greater than or equal to (a >= b)	int, float
Less than (a < b)	int, float
Less than or equal to (a <= b)	int, float

Logical Operations	Applies To
And (a && b)	boolean
Or (a || b)	boolean
Not (!a)	boolean

Matrix Operations	Example
Transpose	matrix1^T

4.4 Functions

For the language, we plan to implement functions as first-class objects. In order to create a function, one must include the word "function" in front of the method. One can see the example in the example program. Once one creates the function, one will be able to pass it as a value.

4.4.1 Example

```
1:  func getDeterminant(Matrix matrix) = findDeterminant;
```

We also plan to add functions to perform row manipulation (i.e. switching rows).

4.5 Comments

In order to comment, one would have to use "//" to make the compiler ignore the line.

4.5.1 Example

```
1:  //Ignore this!
```

5 Example Program

```
1:  function float findDeterminant(Matrix matrix) {
2:      if(matrix.columns != matrix.rows) {
3:          return 0;
4:      }
5:      if(matrix.columns == 1) {
6:          return matrix[0][0];
7:      }
8:      if(matrix.cols == 2) {
9:          return matrix[0][0] * matrix[1][1] - matrix[0][1]
                * matrix[1][0];
10:      }
11:      float determinant = 0;
12:      for(int i = 0; i < n; i++) {
13:          Matrix spliced = matrix.spliceRow(0);
14:          spliced = small.spliceColumn(0);
15:          determinant += (-1)^i * matrix[0][i] *
                findDeterminant(spliced);
16:      }
17:      return determinant;
18:  }
```