**CS323 Documentation**

About 2 pages

**1.** **Problem Statement**

Create a lexical analyzer (LA) for the programming language Rat23S. The LA should be able to read and identify 6 token types (identifiers, keywords, integers, reals, operators, and separators) including any whitespaces. Construct a LA by writing tokens in terms of regular expressions(RE), build a NFSM that validates each token and convert into a DFSM

**2.** **How to use your program**

*<write detailed steps/Instructions how to execute your program>*

e.g.,

Step1) Extract the files from the zip folder.

Step2) Be sure to be in the correct directory with the lexer.exe program.

Step3) (Optional) Add your own code.

* Add a file to ‘testCases’ folder and rename it to ‘code.txt’ In this file any text or code can be added and it will run through our program.
* If there is no code.txt a default will .txt file will be ran a used.

Step4) Run the .exe file through a terminal.

**3.** **Design of your program**

*1)* *Our program will read any user test file and run it through a lexical analyzer*

*2)* *The Lexical analyzer will identify 6 token types (identifiers, keywords, integers, reals, operators, and separators)*

*3)*  *The program has a function that will go through the code give character by character*

*4) It will read through each character until it is recognized by any of the 6 token types*

*6) Then it will assign the item with the corresponding value and type. If it is a space nothing will be outputted*

*7) Our programing will check if the current character is a separator and assign its current item value to the character and its current item type to “Separator”*

*8) Then it will check if the current character value is an operator. It will print out the previous type or set the current type to “Operators”*

*9)Then the program will look for identifiers and keywords. It will check every character that goes through the function and see if it is a keyword. If it is a keyword it will set the type to a “Keyword” and move to the next character. If the new character is a letter then it will remove the “Keyword” type and add the “Identifier” type.*

*10) Next, it will check if the character is a digit and if the last type was a “Identifier” then it will keep it as one, otherwise it will set the value to the number and type to “Real”.*

*11) If there are any spaces they will be ignored and the program will print the current value and type.*

*12) Every time a type changes it will print the value and type before changing it assuming that a new type will be added.*

*13) Everything is printed out line by line and will not print anything until it changes type, which will then print the value before changing the value to the new value and type.*

**4.** **Any Limitation**

*<All features are running according to the assignment but you limit your program due to resource limitations, such as*

*Maximum number of lines in the source code, size of the identifier, integer etc.* ***Say ‘None’ if there is no limitation****>*

*The runtime of our code is one limitation since it does take a few seconds to execute, but that might be solved by optimizing the code and algorithms we are using.*

**5.** **Any shortcomings**

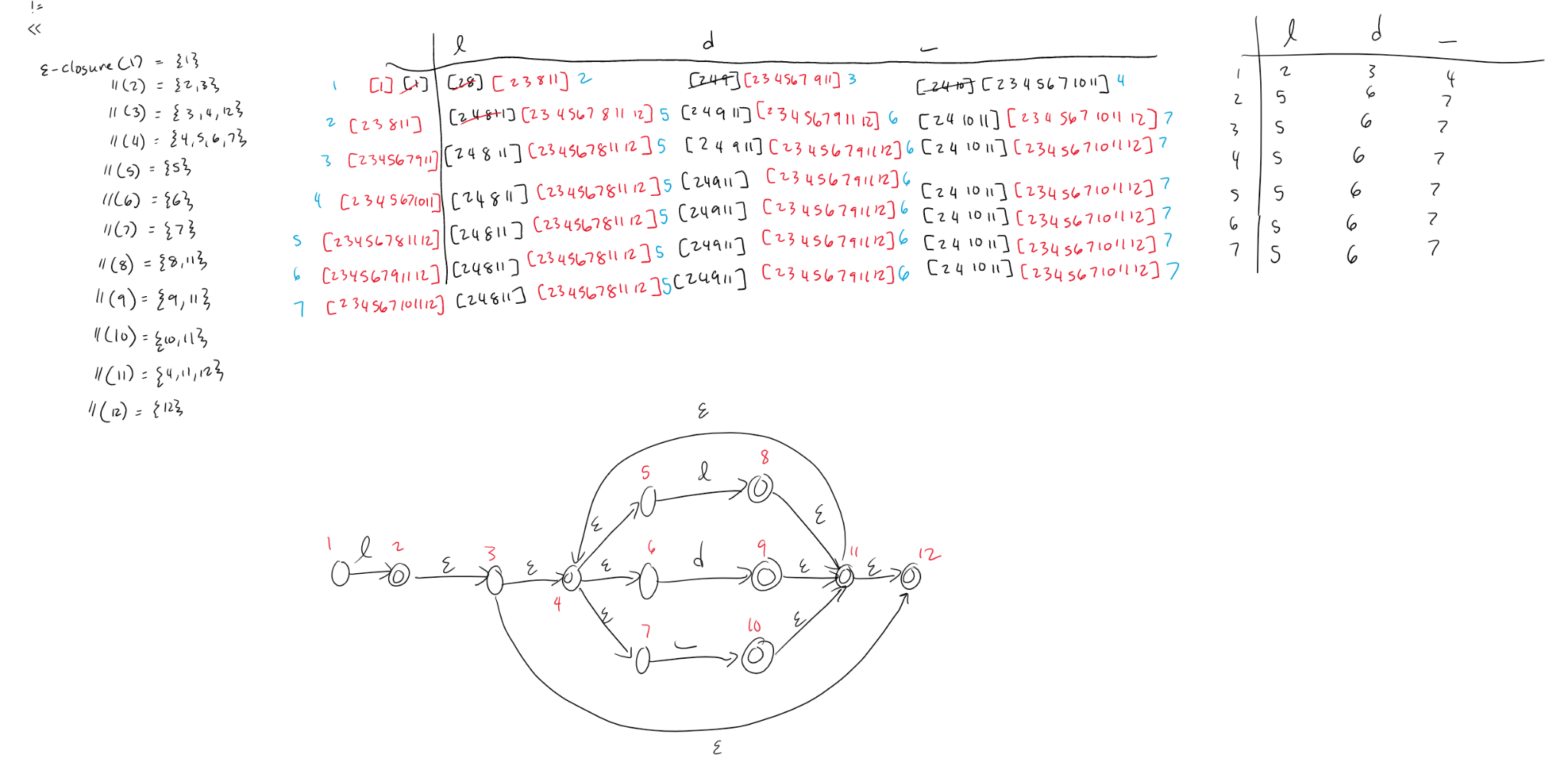
*<Anything you could NOT implement although that is required by the*

*Assignment.* ***Say ‘None’ if there is no shortcoming****>*

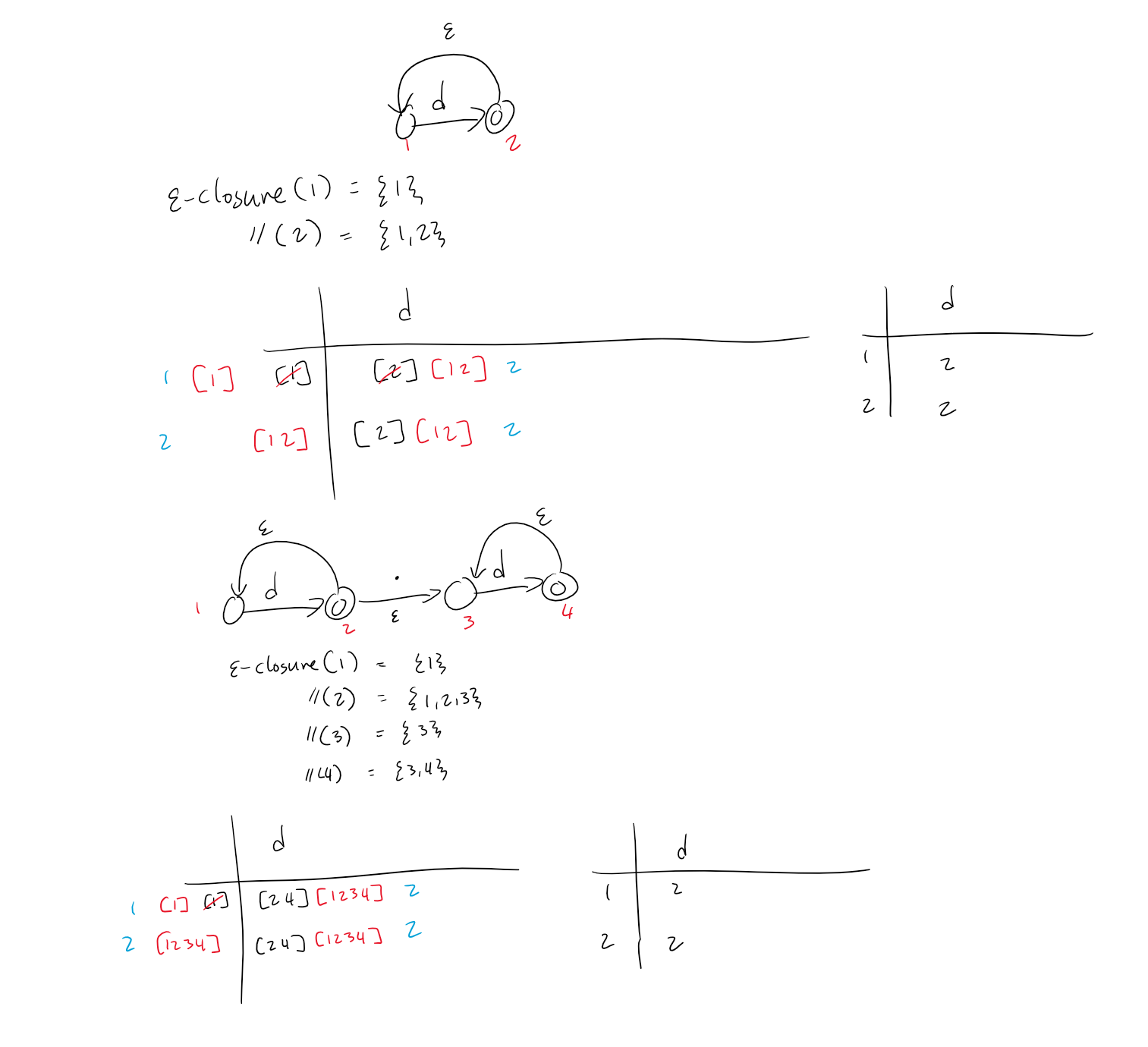
*None*

**Images**

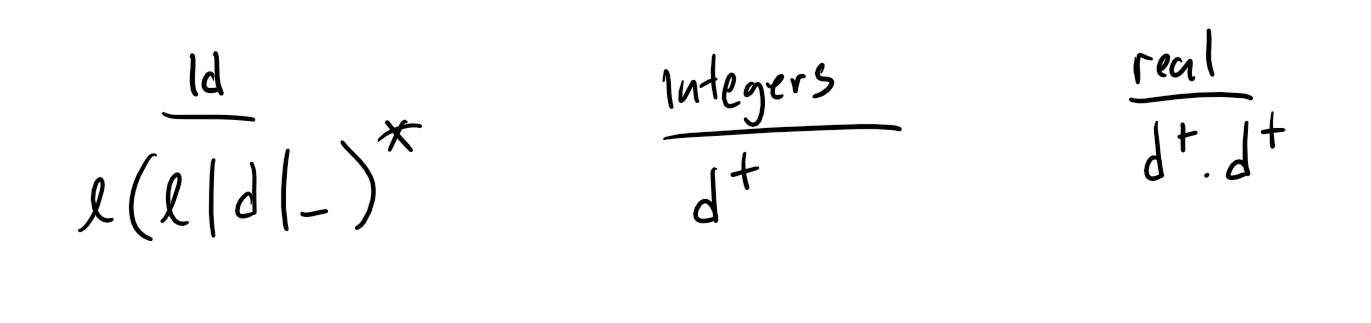
**NFSM-DFSM Identifiers**



**NFSM-DFSM Integers and Reals**



**Regular Expressions**



**Keywords, Operators, and Separators**

