CS323 – Project 1 Documentation

About 2 pages

1. Problem Statement

The purpose of this project is to build a lexical analyzer that uses a Finite State Machine (FSM) to model the possible lexeme categories (tokens) in a programming language. Possible tokens include Keyword, Identifier (Variable), Operator, Real (Number), Separator, Comment and unknown. The program should return the token and the lexeme to be output as a pair for each lexeme instance and write the output to a file.

2. How to use your program

Step 1) go to command line
Step 2) type "`./lexer`"
Step 3) Hit the CR
Step 4) type the `<input filename>`
Step 5) Hit the CR
Step 6) See output at "`out_<input filename>`"

3. Design of your program

State Table (2D array):

Stores the possible output states based on the input state.
Includes:

- INTEGER (`564`)
- REAL (`3.14`)
- OPERATOR (+)
- STRING
- SPACE
- SEPARATOR (`{`)
- COMMENT (`!  !`)
- UNKNOWN (`3.14.65`)

Uses a map to store possible keywords to check against a string.

Token Types:

Any strings that are not a keyword are identifiers.

Any numbers that contain only one "." are Reals.

Any numbers that are only numeric are Integers.

Separators and Operators are explicitly defined as their characters as shown in the language file.

Anything that doesn't meet the above criteria is considered an Unknown.

4. Any Limitation

- Character set is ASCII only (no Unicode languages)
  - a-z, A-Z (no accents)
- Real (Numbers) formatting is defined using the American standard using `.` for decimals.
  - `,` is not considered valid within a Real.

5. Any shortcomings

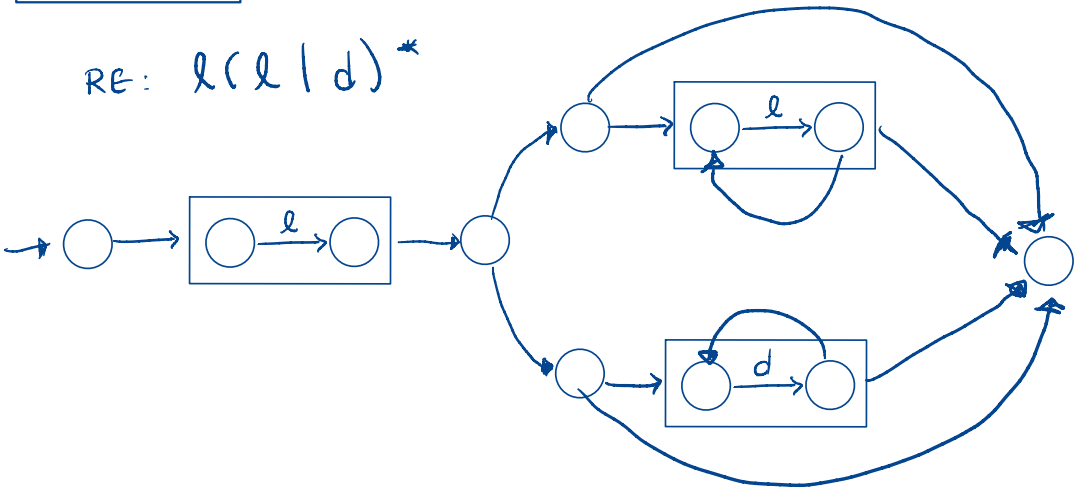
- Block comments spanning multiple lines are not working.
- Spacing is required around all operators "`i = 3`" vs "`i=3`"
- IDENTIFIERS starting with a numeric are split
  - "`1string`"
    - 1 – integer
    - String - identifier

$l = \Sigma$ a-z, A-Z

$d = \Sigma$ 0-9

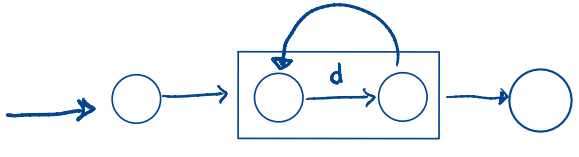
Identfiers

RE: $l(l \mid d)^*$



Integers

RE: d^+



Reals

RE: $d^*(.)\,d^+$