|  |
| --- |
| Team 10 |
| **Compiler Term-Project #1** |
| The implementation of a lexical analyzer |

|  |  |
| --- | --- |
| Date | May 9, 2020 |
| Instructor | Hyosu Kim |
| Team | #10 |
| Members | Heesang Ro (20145001)  Junhyuck Woo (20145337) |



# INDEX

[INDEX 1](#_Toc39255872)

[LEXICAL SPECIFICATIONS 2](#_Toc39255873)

[TOKENS 3](#_Toc39255874)

[REGULAR EXPRESSION 4](#_Toc39255875)

[NFA (Non-deterministic Finite Automata) 5](#_Toc39255876)

[1. ID 5](#_Toc39255877)

[2. INTEGER 6](#_Toc39255878)

[3. FLOAT 8](#_Toc39255879)

[4. LITERAL 11](#_Toc39255880)

[DFA (Deterministic Finite Automata) 12](#_Toc39255881)

[1. ID 12](#_Toc39255882)

[2. INTEGER 14](#_Toc39255883)

[3. FLOAT 15](#_Toc39255884)

[4. LITERAL 17](#_Toc39255885)

[LIMITATION 19](#_Toc39255886)

[SOLUTION 19](#_Toc39255887)

[IMPLEMENTATION 20](#_Toc39255888)

[1. Chess System 20](#_Toc39255889)

[2. Hardware (Robot Arm) 20](#_Toc39255890)

[3. Network 20](#_Toc39255891)

[4. Computer Vision 20](#_Toc39255892)

[TEST CASES & RESULT 21](#_Toc39255893)

[APPENDIX 22](#_Toc39255894)

[1. NFA to DFA with transition table 22](#_Toc39255895)

# LEXICAL SPECIFICATIONS

|  |
| --- |
| **Variable type**   * int fir a signed integer * char for a literal string * bool for a Boolean string * float for a floating-point umber   **Signed integer**   * A single zero digit * A non-empty sequence of digits, starting from a non-zero digit * A non-empty sequence of digits, starting from a minus sign symbol and a non-zero digit   **Literal string**   * Any combination of digits, English letters, and blanks, starting from and terminating with a symbol “   **Boolean string**: true and false  **Floating-point number**   * A sequence that meets the following conditions:  1. It starts with or without a negative sign symbol 2. . (a decimal point) appears only once 3. Scientific/exponential symbols like E are not allowed 4. Both left and right side of the decimal point must not be empty sequence 5. The left side of a decimal point must be a single digit 0 or a non-empty sequence starting from a non-zero digit 6. The right side of a decimal point must be a single digit 0 or a non-empty sequence terminating with a non-zero digit   **An identifier of variables and functions**   * A non-empty sequence of English letters, digits, and underscore symbols, starting from an English letter or a underscore symbol   **Keywords for special statements**   * if for if statement * else for else statement * while for while-loop statement * for for for-loop statement * return for return statement   **Arithmetic operators**: +, -, \*, and /  **Bitwise operator**: <<, >>, &, and |  **Assignment operator**: =  **Comparison operators**: <, >, ==, !=, <=, and >=  **A terminating symbol of statements**: ;  **A pair of symbols for defining area/scope of variables and functions**: { and }  **A pair of symbols for indicating a function/statement**: ( and )  **A symbol for separating input arguments in functions**: ,  **Whitespaces**: a non-empty sequence of \t, \n, and blanks |

# TOKENS

|  |  |
| --- | --- |
| Token | Lexeme |
| VARIABLE | int, float, char, bool |
| KEYWORD | if, else, while, for, return |
| LOGIC | true, false |
| ID | i, j, k, ab\_123, func1, func\_, \_\_func\_bar\_\_, …… |
| INTEGER | 0, 1, 22, 123, 56, -1, -22, -123, -56, …… |
| FLOAT | 0.5, 0.0, -10.0, 100.00001, ,,,,, |
| LITERAL | “Hello world”, “My student id is 12345678”, …… |
| OPERATOR | +, -, \*, /, <<, >>, &, | |
| COMPARISON | <, >, ==, !=, <=, >= |
| WHITESPACE | , \t, \n |
| BRACE | {, } |
| PAREN | (, ) |
| ASSIGN | = |
| TERM | ; |
| COMMA | , |

# REGULAR EXPRESSION

Alphabet (Σ)

* A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, \*, /, <, &, |, =, ;, , {, }, (, ), ., ,

Symbol

* zero = 0
* non-zero = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
* letter = A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z

|  |  |
| --- | --- |
| Token | Regular Expression |
| ID |  |
| INTEGER |  |
| FLOAT |  |
| LITERAL |  |
| VARIABLE |  |
| KEYWORD |  |
| LOGIC |  |
| OPERATOR |  |
| COMPARISON |  |
| WHITESPACE |  |
| BRACE |  |
| PAREN |  |
| ASSIGN | = |
| TERM | ; |
| COMMA | , |

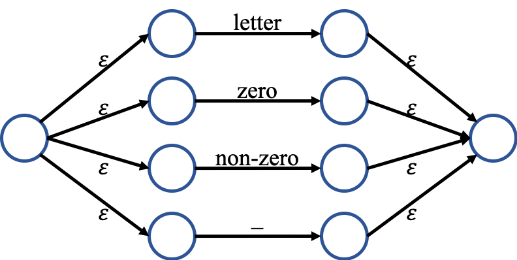
# NFA (Non-deterministic Finite Automata)

Our team implemented the NAF (Non-deterministic Finite Automata) with the McNaughton-Yamada-Thompson algorithm. We only handle 4 kinds of tokens, and the other tokens will be implemented as code without a transition table.

## 1. ID

장치이(가) 표시된 사진

자동 생성된 설명



검은색, 브러시이(가) 표시된 사진

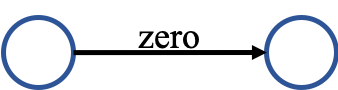
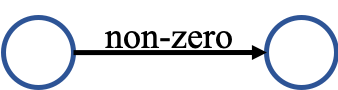
자동 생성된 설명

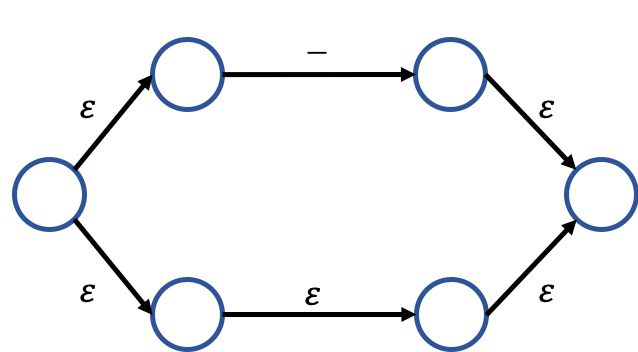
브러시이(가) 표시된 사진

자동 생성된 설명

## 2. INTEGER

* ,



장치이(가) 표시된 사진

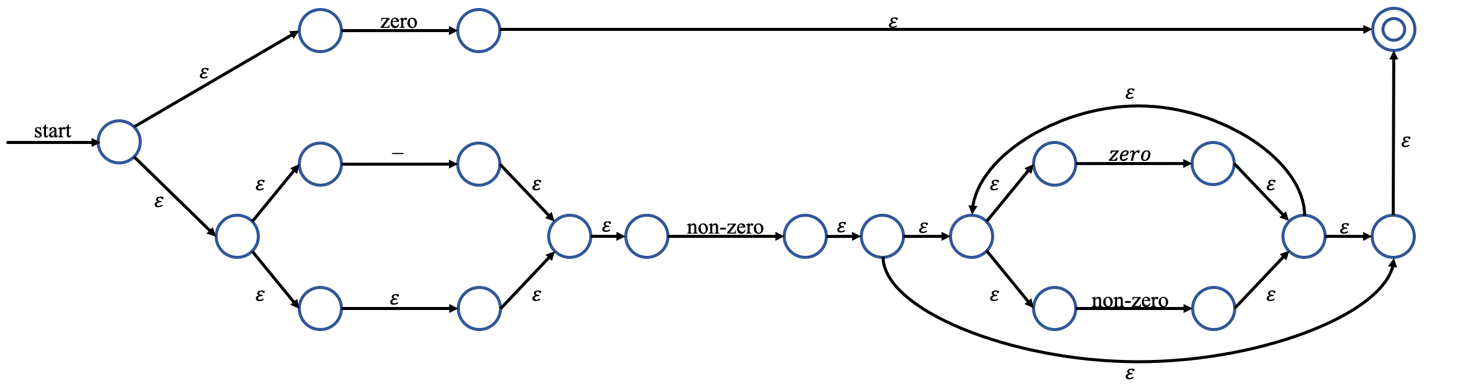
자동 생성된 설명

검은색이(가) 표시된 사진

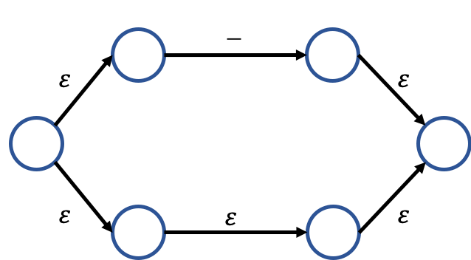
자동 생성된 설명

시계이(가) 표시된 사진

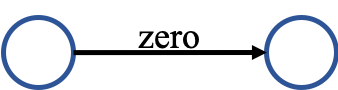
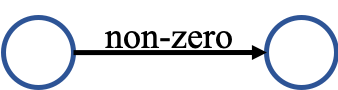
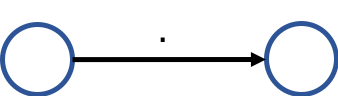
자동 생성된 설명



## 3. FLOAT



* , .

장치이(가) 표시된 사진

자동 생성된 설명

검은색이(가) 표시된 사진

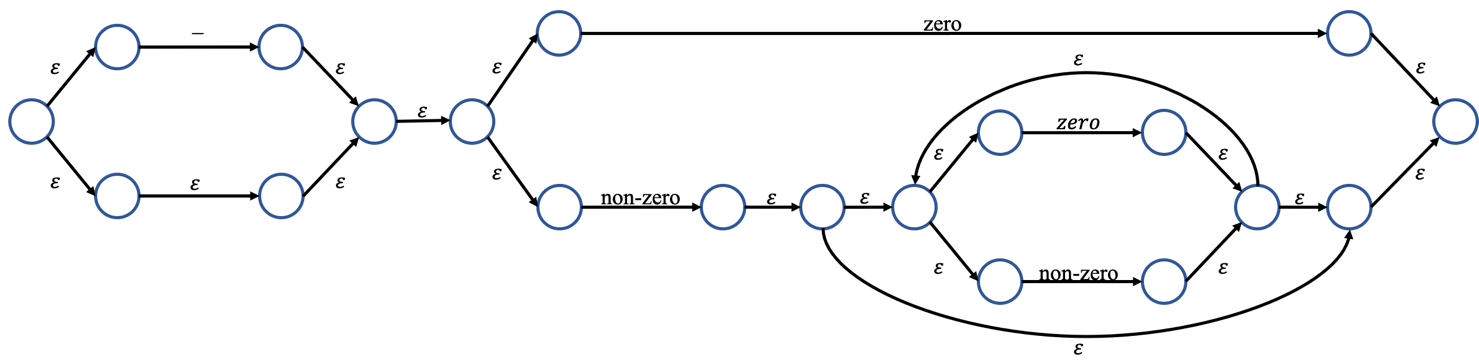
자동 생성된 설명

검은색, 보는, 앉아있는, 서있는이(가) 표시된 사진

자동 생성된 설명

검은색이(가) 표시된 사진

자동 생성된 설명



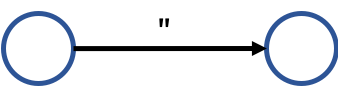
검은색, 보는, 대형, 앉아있는이(가) 표시된 사진

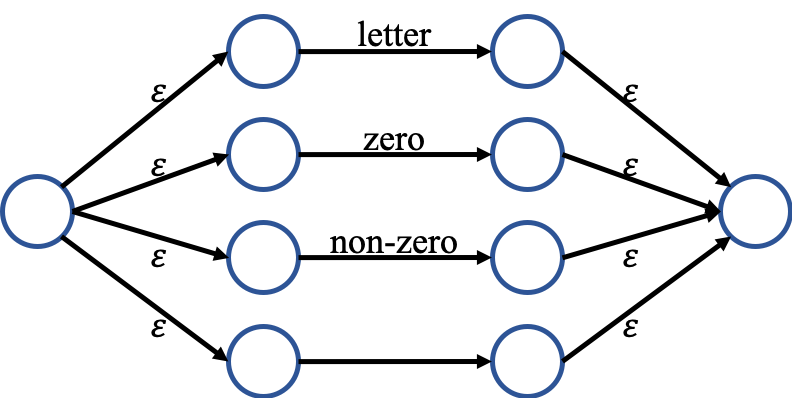
자동 생성된 설명

주방용품, 강판, 여과기, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

## 4. LITERAL





브러시이(가) 표시된 사진

자동 생성된 설명

브러시이(가) 표시된 사진

자동 생성된 설명

# DFA (Deterministic Finite Automata)

We built DFA graph and transition table using subset (powerset) construction algorithm.

If you want to see the hand-writing paper, please check the appendix.

## 1. ID

모니터, 검은색, 화면, 텔레비전이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| = {A, B, C} = T0 |
| = = {D, F, G, H, I, J, K, L, R} = T1  =  =  = = {E, F, G, H, I, J, K, L, R} = T2 |
| = = {M, Q, R, H, I, J, K, L} = T3  = = {N, Q, R, H, I, J, K, L} = T4  = = {O, Q, R, H, I, J, K, L} = T5  = = {P, Q, R, H, I, J, K, L} = T6 |
| = = T3  = T4  = = T5  = = T6 |
| = = T3  = T4  = = T5  = = T6 |
| = = T3  = T4  = = T5  = = T6 |
| = = T3  = T4  = = T5  = = T6 |
| = = T3  = T4  = = T5  = = T6 |

* Transition table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | letter | zero | non-zero | \_ |
| * T0 | T1 |  |  | T2 |
| *T1* | T3 | T4 | T5 | T6 |
| *T2* | T3 | T4 | T5 | T6 |
| *T3* | T3 | T4 | T5 | T6 |
| *T4* | T3 | T4 | T5 | T6 |
| *T5* | T3 | T4 | T5 | T6 |
| *T6* | T3 | T4 | T5 | T6 |

* DFA Graph

검은색, 옅은, 빨간색, 얼굴이(가) 표시된 사진

자동 생성된 설명

## 2. INTEGER

텔레비전, 화면, 모니터, 검은색이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| = {A, B, C, D, E, G, H, I} = T0 |
| = = {F, H, I} = T1  = = {T, S} =T2  = = {J, K, L, M, N, R, S} =T3 |
| =  =  = = T3 |
| =  =  =  = |
| = {O, Q, L, M, N, R, S} = T4  = = {P, Q, L, M, N, R, S} = T5 |
| =  = T4  = = T5 |
| =  = T4  = = T5 |

* DFA Graph
* 실외, 잔디, 빨간색, 평야이(가) 표시된 사진

  자동 생성된 설명Transition table

|  |  |  |  |
| --- | --- | --- | --- |
|  | - | zero | non-zero |
| * T0 | T1 | T2 | T3 |
| T1 |  |  | T3 |
| *T2* |  |  |  |
| *T3* |  | T4 | T5 |
| *T4* |  | T4 | T5 |
| *T5* |  | T4 | T5 |

## 3. FLOAT

컴퓨터, 검은색, 옅은, 노트북이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| = {1, 2, 3, 4, 6, 7, 8, 10} = T0 |
| = = {5, 6, 7, 8, 10} = T1  = = {9, 20, 21} =T2  = = {11, 12, 13, 14, 16, 19, 20, 21} =T3  = |
| =  = = T2  = = T3  = |
| =  =  =  = = {22, 23, 24, 30, 31, 32, 34, 37, 38} = T4 |
| =  = {13, 14, 15, 16, 18, 19, 20, 21} = T5  = {13, 14, 16, 17, 18, 19, 20, 21} = T6  = = T4 |
| =  = {25, 31, 32, 33, 34, 36, 37, 38, 40} = T7  = = {31, 32, 34, 35, 36, 37, 38, 39, 40} = T8  = |
| =  = T5  = = T6  = = T4 |
| =  = T5  = = T6  = = T4  = |

= {31, 32, 33, 34, 36, 37, 38} = T9

|  |
| --- |
| = = T8  = |
| =  = T9  = = T8  = |
| =  = T9  = = T8  = |

* Transition table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | - | zero | . | non-zero |
| * T0 | T1 | T2 |  | T3 |
| T1 |  | T2 |  | T3 |
| T2 |  |  | T4 |  |
| T3 |  | T5 | T4 | T6 |
| T4 |  | T7 |  | T8 |
| T5 |  | T5 | T4 | T6 |
| T6 |  | T5 | T4 | T6 |
| T7 |  | T9 |  | T8 |
| T8 |  | T9 |  | T8 |
| T9 |  | T9 |  | T8 |

* DFA Graph

검은색, 샷, 어두운, 잔디이(가) 표시된 사진

자동 생성된 설명

## 4. LITERAL

텔레비전, 화면, 검은색, 모니터이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| = {A} = T0 |
| = = {B, C, D, E, F, G, H, N, O} = T1  =  =  =  = |
| = = {P} = T2  = = {I, M, N, O, D, E, F, G, H} = T3  = = {J, M, N, O, D, E, F, G, H} = T4  = = {K, M, N, O, D, E, F, G, H} = T5  = = {L, M, N, O, D, E, F, G, H} = T6 |
| =  =  =  =  = |
| = = T2  = = T3  = T4  = = T5  = = T6 |
| = = T2  = = T3  = T4  = = T5  = = T6 |
| = = T2  = = T3  = T4  = = T5  = = T6 |
| = = T2  = = T3  = T4  = = T5  = = T6 |

* Transition table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | “ | letter | zero | non-zero |  |
| * T0 | T1 |  |  |  |  |
| T1 | T2 | T3 | T4 | T5 | T6 |
| *T2* |  |  |  |  |  |
| T3 | T2 | T3 | T4 | T5 | T6 |
| T4 | T2 | T3 | T4 | T5 | T6 |
| T5 | T2 | T3 | T4 | T5 | T6 |
| T6 | T2 | T3 | T4 | T5 | T6 |

* DFA Graph

검은색, 빨간색, 잔디, 화면이(가) 표시된 사진

자동 생성된 설명

# LIMITATION

- Size of the chess board is dependent on the size of the robot arm because the robot arm cannot handle whole locations of the board if the board size is too big or small.

- Although there are plenty of special rules for chess, we do not consider any special rules in the chess system.

# SOLUTION

- Size of the chess board is dependent on the size of the robot arm because the robot arm cannot handle whole locations of the board if the board size is too big or small.

- Although there are plenty of special rules for chess, we do not consider any special rules in the chess system.

# IMPLEMENTATION

This project is comprised of mainly four parts: chess system, hardware (robot arm), network, computer vision.

## 1. Chess System

- We will implement this system using C++ language. Models for the system will be designed through many classes. They can stand for pieces, board, rules, etc. Any required information to store will be presented through these classes. Views, which will be implemented using the Qt framework, will display which pieces are located on the chess board now. The views are not related to the chess game directly, but it can show us what’s going on.

▲ Flow of the chess system working

## 2. Hardware (Robot Arm)

- We will use hardware specifically robot arm. This robot will have 5 Degree of freedom which includes gripper. This robot is mainly designed for move pieces of chess. The robot arm will be moved when it gets information from the network. Each of the motors will be controlled properly depending on the data such as moving location and piece type.

▲ Expected robot arm and chess piece

## 3. Network

- We will implement this system using C++ language, using the boost’s Asio socket library. It will use a specific protocol for communicating between a robot arm and a chess system. This system consists of sending and receiving. The receiving part should open a socket and listen to messages, then the sending part connects to it and sends data. In this project, the receiving part will be a robot arm, and the sending part will be a camera. They can communicate some location information through this system.

▲ Data flow from a camera to a system, using the network

▲ Data flow from a system to a robot arm, using the network

## 4. Computer Vision

- We use the OpenCV for computer vision. After taking pictures of a chessboard, we recognize each cell of the grid pattern as coordinate. And then we determine if a chess piece on each coordinate using object recognition. Finally, we discern the difference with two pictures that previous and recent ones to recognize the movement of the chess piece.

▲ Recognize coordinate

▲ Flow of the computer vision

# TEST CASES & RESULT

- Size of the chess board is dependent on the size of the robot arm because the robot arm cannot handle whole locations of the board if the board size is too big or small.

- Although there are plenty of special rules for chess, we do not consider any special rules in the chess system.

# APPENDIX

## 1. NFA to DFA with transition table