Compiler term project

Implementation of lexical analyzer

Class by prof. Hyosu Kim

20184395 Myeongwon Choi

20186491 Chihyun Song

Table of Contents

[Abstract 3](#_Toc39447002)

[Language specification 3](#_Toc39447003)

[Regular expression definition 3](#_Toc39447004)

[NFA and DFA definition 4](#_Toc39447005)

# Abstract

We implemented a syntax analyzer by implementing a DFA class which acts based on the definition introduced on compiler lecture.

And we generated instances of it and defined its behavior based on DFA detail we generated.

# Language specification

## Regular expression definition

First we made regular expressions of given tokens so we can make DFAs with it.

* (positive\_digit): (1|2|3|4|5|6|7|8|9)
* (digit): (0|1|2|3|4|5|6|7|8|9)
* (alphabet): (a|b|c|…|z|A|B|…|Z)

1. Signed integer(INT): ((-|ε)(positive\_digit)(digit\*)|0)
2. Literal string(STRING): “(alphabet| |digit)”
3. ~~Floating-point number(FLOAT): (-|ε)(positive\_digit)(digit\*).(digit\*)~~
4. An identifier of variables and functions(ID): (\_|alphabet)(alphabet|digit|\_)\*
5. Whitespaces(WHITESPACE): (\t| |\n)\*
6. Variable type(TYPE),

Boolean string(BOOLEAN),

Keywords for special statements(KEYWORD),

Arithmetic operators(ARITHMETIC),

Bitwise operators(BITWISE),

Assignment operator(ASSIGNMENT),

Comparison operators(COMPARISON),

A terminating symbol of statements(SEMICOLON),

A pair of symbols for defining area/scope of variables and functions(BRACE),

A pair of symbols for indicating a function/statement(PARENTHESES),

A symbol for separating input arguments in functions(SEPARATOR)

: all same. Just write all the accepted lexemes and separate them with |.

Actually, most of token definitions are just based on a list of words accepted. So, there is some redundant token definitions that is totally clear. We skipped that kind of token definition.

## NFA and DFA definition

The strict type of DFA must have exactly one transition for one input alphabet for all current state. So DFA follows strict definition has some trap state which cannot derive to final state to represent the halted state of DFA.

But we don’t have to implement our DFA in this way, our casual definition of a DFA will have at most one possible transition for one input alphabet for all current state. If a DFA gets an input alphabet which is not defined as an alphabet, The DFA will simply halt at that moment.

Although we should follow the procedure which converts a regular expression into a NFA and translate it again into a DFA, but we decided to build DFA directly since the language specification is simple so we can make efficient DFA directly.

* Pos:1|2|3|4|5|6|7|8|9
* Dig:1|2|3|4|5|6|7|8|9|0
* Alpha:a|b|c|…|z|A|B|…|Z

INT

A close up of text on a white background

Description automatically generated

STRING

A close up of text on a white background

Description automatically generated

FLOATA close up of text on a white background

Description automatically generated

IDA close up of text on a black background

Description automatically generated

WHITESPACEA close up of text on a black background

Description automatically generated

# Program implementation

We defined corresponding DFA. In order to get things work, we need to implement DFA class works based on our DFA designs.

First, a DFA consists of:

* a finite set of states
* a finite set of input symbols called the alphabet
* a transition function
* an initial or start state
* a set of accept states

Among the formal definition of DFA, we need the information of alphabet, transition functions, initial state, and a set of final states. Actually, an information about set of state can be omitted because unreachable sets are useless. And the object definition of DFA has one more field, halted or running.

So constructor will simply get mentioned values, and initialize the field values with input.

Next, we have process function which gets a character or string input and process it. Giving an automaton object string input is equivalent to giving character input sequentially.

First, automaton checks if input character is defined as alphabet. If not, DFA will halt.

And check if the automata is already halted. If so, it will simply do nothing because halted automaton does not accept any character.

Finally a DFA will traverse around transition function list to find a transition function which matches the current state and the input character. If found, change current state. If not found, it means transition function not found. halt DFA.

And a DFA has additional operation which real world DFA does not have, setState. It directly sets current state with input parameter, and it will set DFA state to running.