Finite Automaton

Student: Alexandrescu Andrei-Robert, 931/1

Current lab: Repository Link

Integration with previous lab: Repository Link

1. File Structure (FA.in)

a) Mathematic (Natural Language) description

s_1 s_2 ... s_n (states)
a_1 a_2 ... a_m (alphabet)
noTran (number of transitions)
p_1 b_1 q_1 (delta(p_1, b_1) = q_1)
...
p_{noTran} b_{noTran} q_{noTran}
q_0 (initial state)
f_1 f_2 ... f_o (final states)

b) EBNF:

identifier ::= letter | letter {letter | digit}

letter ::= "A" | "B" | . ..| "Z"

digit ::= "0" | "1" |...| "9"

non_zero_digit ::= “1” | ... | “9”

constno ::= [(“+” | “-“)] non_zero_number | zero

zero ::= 0

non_zero_number::= non_zero_digit{digit}

states ::= identifier | identifier {identifier}

alphabet ::= { constno }

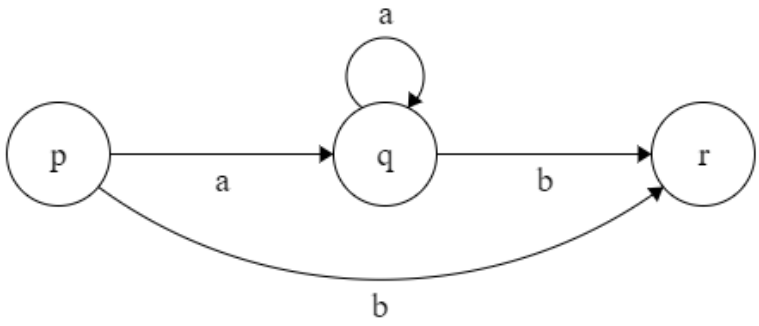
transitions ::= {identifier constno identifier}

initialState ::= identifier

finalStates ::= identifier {identifier}

c) Example:

p q r
a b
4
p a q
q a q
q b r
p b r
p
r



2. Program details

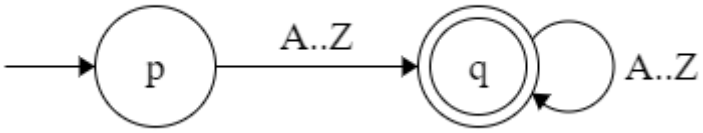
Method *readFA* is used to read the data from the *FA.in* file and store it accordingly in the RAM. Some error cases are treated such as:

- one of the transition terms (state 1, transition term, state 2) does not belong to the declared states / alphabet respectively
- the initial state does not belong to the declared states
- one of the final states does not belong to the declared states

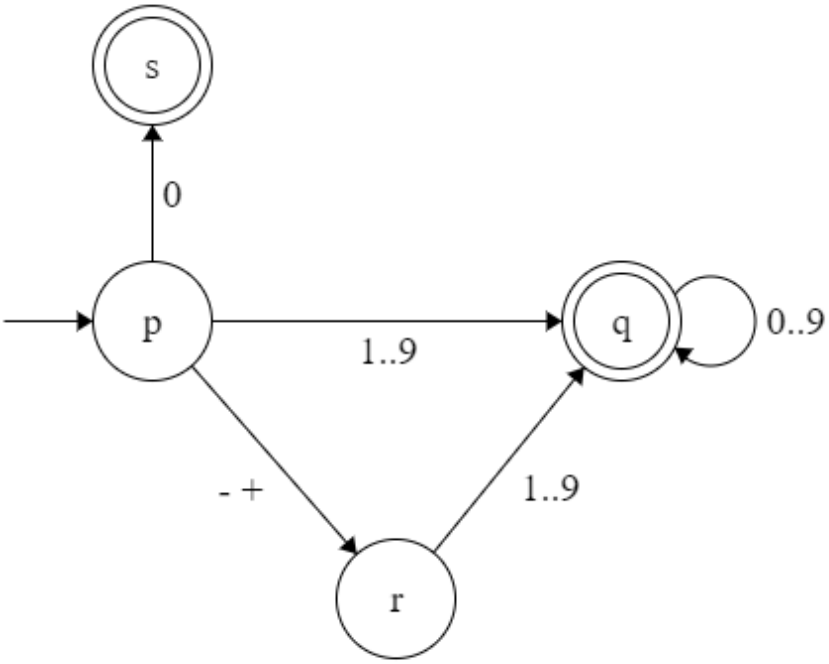
Method *verifySequence* checks whether a given sequence is accepted by the FA. This is done by simply using a for loop to cycle through the characters of the sequence and using a *currentState* variable to keep track of the current state. The method *move* is used to transition between states using the current symbol from the alphabet. In case the sequence could not be consumed entirely, an error occurs.

3. Integration with labs 1-3

Two finite automations were created for identifying identifiers and constants. The first one, for identifiers, uses the *FA_identifiers.in* description of a finite automation. The corresponding drawing can be seen below:



The second finite automaton is read from the *FA_constants.in* file corresponding to the following finite automaton:



I have added the method *identifyIdentifiersConstants* to the *Compiler* class in which all elements from the symbol table are considered. For each element, we check if it is either an identifier (*getIsIdentifier*) or a numerical constant (*canBeNumber*). If the element is an identifier it is considered as a sequence for the Identifiers Finite Automaton (called *parserIdentifiers*) and the *verifySequence* method is called. If the element is a numerical constant, it becomes a sequence for the Constants Finite Automation (called *parserConstants*). The verifySequence method is called as well here. Asserts are used to ensure the correctness of the program.