Compiler Project

Phase1: Lexical Analyzer Generator

Team members

Aya Lotfy

Salma Mohamed

Salma Yehia

Mohamed Gaber

Data Structure

3D vector of integer

To represent NFA and DFA
Each entry in this 3D vector contains a source state.
Each state is represented by 2D vector where each entry contains input (can the state make a transition under it) and target state (which can be arrived under the same input)

Vector of the accepted states

Each entry contains the number of the state and its name.

Vector of Variables

It contains the state machine of each variable so it can be easily included in any expression.

Algorithms and Techniques

Reading rules File and converting it to NFA

The program starts by scanning rules then detect the type of this rule as regular definition, regular expression, keywords or punctuation.

In case of regular definition and expressions the rule is converted from infix to postfix then evaluated using stack then creates a state machine for this rule.

In case of regular definition the state machine is added as a variable in the variables vector.

In case of regular expression the state machine is added to the NFA with the starting state pointing to its starting state and its name is added to the accepted states.

Convert NFA to DFA

After constructing the NFA, DFA is generated by looping on the states and identifying redundant states.

DFA Minimization

First the code separates acceptance states -based on accepted string- from non-acceptance states then assign numbers to each set created.

The main loop iterates over the current set size then for each set member it’s compared with all other elements if it matches transitions with other state it’s removed from the set and added to a new set.
After creating the final minimal sets it takes the first element of each set as a representative for that set then it assigns new numbering for the minimized states.
Than iterate over the original DFA transition table replacing each state with its representative ignoring the states which are handled by their representatives then it returns the constructed minimized DFA table and minimized states.

Recognize tokens

Here, the program reads the code file and then divides it line by line, and then according to the input it visits nodes in the DFA graph until it find a space. If it is not an accepted state generate error message. If it is an accepted state write to the tokens file the token.

Assumptions

1. One token cannot be divided by space.

So it is not confusing.

2. Only regular definitions cannot be included in any other expression.

Regular definitions is treated as variables so they can be included in any other expression

3. Any operator used as operand is preceded by escape backslash character

So the program wouldn't be confused if it is operator or operand.