Compilers project phase 1 report

# Data structures & algorithms used

## state class

A class implementation of a state in the state machine.

### members

* m\_Id: a unique identifier for each state
* m\_Transitions: a map of transitions to other states given a certain character as input

### methods

* addTransition: a method to add a transition from this state to another given an input
* addTransitions: a method to add a transition from this state to a list of states given an input

## nfa class

A class implementation of a non-deterministic finite state machine.

### members

* m\_StartState: the starting state of the NFA
* m\_TerminalStates: the set of terminal states of the NFA

### methods

* concatenate: a method to concatenate an NFA to this NFA using Thompson’s method where all the transitions from the second NFA’s start state are set as valid transition to the first NFA’s combined terminal state.
* unionize: a method to create a union of an NFA with this NFA using Thompson’s method where a new starting state is created with an epsilon transition to both NFAs start states and a new terminal state is created with an epsilon transition from both NFAs terminal states.
* kleeneClosure: a method to create a kleene closure of this NFA using Thompson’s method where new starting and terminal states are created. An epsilon transition is created from the new start state to the original start state and the new terminal state. Another epsilon transition is created from the original terminal state to the original start state.
* positiveClosure: a method to create a positive closure of this NFA using Thompson’s method where new starting and terminal states are created. An epsilon transition is created from the new start state to the original start state. Another epsilon transition is created from the original terminal state to the original start state.

## DFA class

A class implementation of a deterministic finite state machine.

### members

* m\_StartClosure: the epsilon closure of the starting state of the corresponding NFA
* m\_TerminalClosures: the epsilon closure of the terminal states of the corresponding NFA
* m\_Table: the table of transitions of each epsilon closure in the DFA
* m\_NFATerminalStates: the set of terminal states of the corresponding NFA

### methods

* getEpsilonClosure: a method to get the epsilon closure of a given state using depth first search on the transition graph adding to a set the states that can be reached using epsilon transitions.
* subsetConstruction: a method to convert an NFA to a DFA using the subset construction algorithm. A depth first search is performed starting from the epsilon closure of the start state of the NFA where given a certain input, all reachable states (epsilon closures) from this epsilon closure is added to a transition table and then DFS is performed again from the reachable states until all possible epsilon closures are visited.
* initializePartitions: a method that initialises the partitions in the minimisation of the DFA where the starting partitions contain a partition for all non-terminal states and a partition for each terminal state in the m\_NFATerminalStates.
* splitPartition: a method that splits each partition according to the transitions of the states given a certain input. For each state in the partition a set of transitions is created. The set of states corresponding to each unique set of transitions are split from the original partition and put in a new partition.
* minimize: a method that minimises the DFA into a corresponding minimal DFA. The DFA states are partitioned into initial partitions using initializePartitions method. The partitions are continuously attempted to be split into smaller partitions using splitPartition method until no partition can be split. The new partitions are now the new states of the minimised DFA, and their transitions are copied over to m\_Table.

## regex handler

### methods

## lexical analyzer generator class

### members

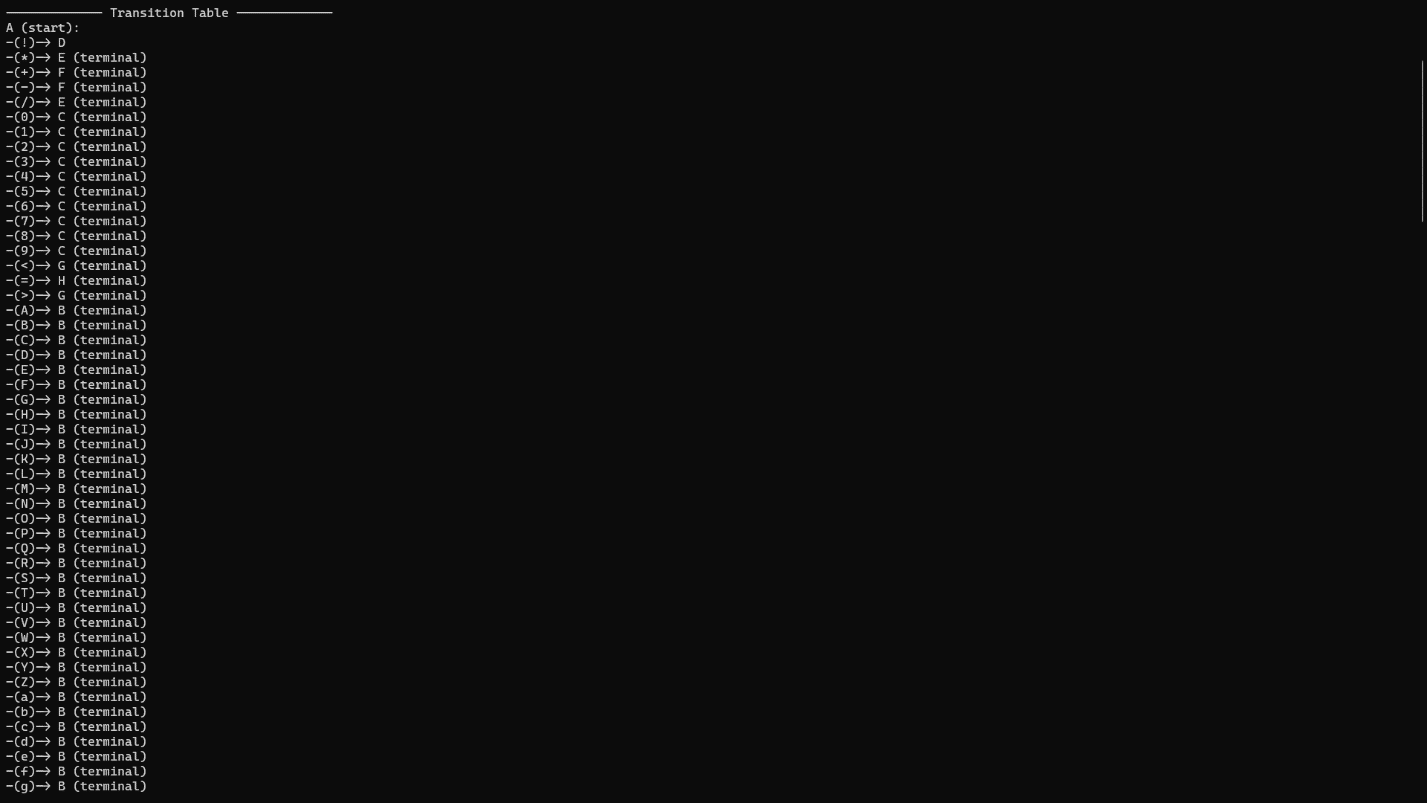
### methods

## lexical analyzer class

### members

### methods

# resultant transition table for the minimal DFA



# resultant stream of tokens for the example test program

