**FLCD Lab. 4**

**- documentation -**

**GitHub Repository**:

<https://github.com/adrianPascan/FLCD_Lab2>

**FA (Finite Automaton)**

1. **input file format**

For a FA M=(Q,Σ,δ,q0,F),

STATE ϵ Q

SYMBOL ϵ Σ

file := states “\n” alphabet “\n” transitions “\n” initialState “\n” finalStates

states := STATE | STATE “|” states

transitions := transition | transition “|” transitions

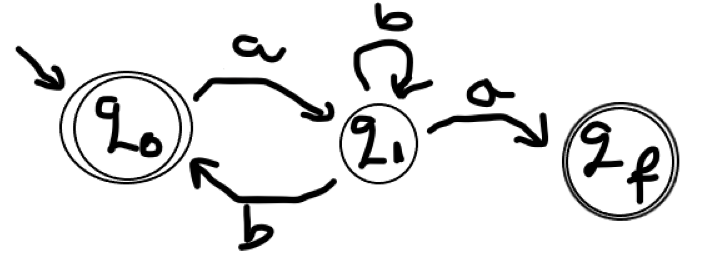
transition := STATE “,” SYMBOL “,” STATE

initialState := STATE

finalStates := finalState | finalState “|” finalStates

finalState := STATE

1. **example**



*fa.in*:

q0|q1|qf  
a|b  
q0~a~q1|q1~a~qf|q1~b~q0|q1~b~q1  
q0  
q0|qf

*FA object*:

FA(

states=[q1, qf, q0],

alphabet=[a, b],

transitions={

TransitionInput(state=q1, symbol=b)=TransitionOutput(states=[q1, q0]),

TransitionInput(state=q1, symbol=a)=TransitionOutput(states=[qf]),

TransitionInput(state=q0, symbol=a)=TransitionOutput(states=[q1])

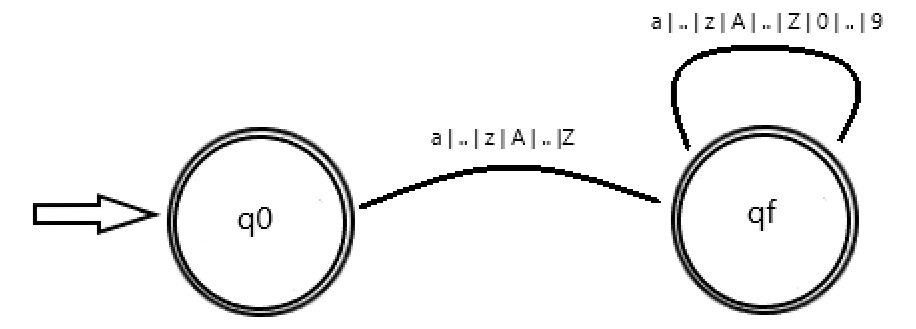
},

initialState=q0,

finalStates=[qf, q0]

)

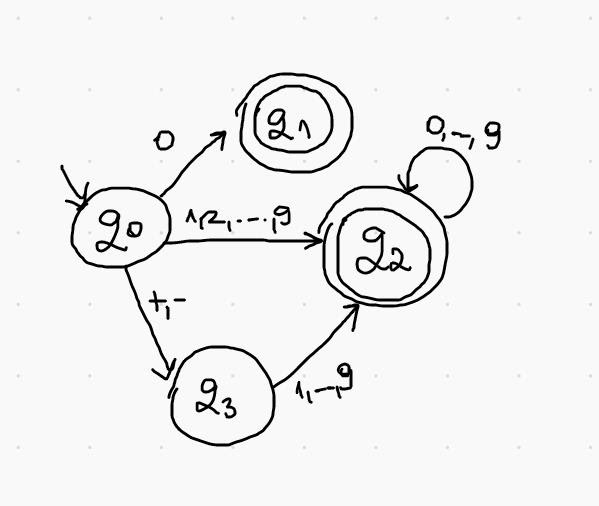
1. **FA for identifier**

****

*fa\_iden.in:*

1: q0|q1  
2: 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  
3: q0~a~q1|q0~b~q1|q0~c~q1|q0~d~q1|q0~e~q1|q0~f~q1|q0~g~q1|q0~h~q1|q0~i~q1|q0~j~q1|q0~k~q1|q0~l~q1|q0~m~q1|q0~n~q1|q0~o~q1|q0~p~q1|q0~q~q1|q0~r~q1|q0~s~q1|q0~t~q1|q0~u~q1|q0~v~q1|q0~w~q1|q0~x~q1|q0~y~q1|q0~z~q1|q0~A~q1|q0~B~q1|q0~C~q1|q0~D~q1|q0~E~q1|q0~F~q1|q0~G~q1|q0~H~q1|q0~I~q1|q0~J~q1|q0~K~q1|q0~L~q1|q0~M~q1|q0~N~q1|q0~O~q1|q0~P~q1|q0~Q~q1|q0~R~q1|q0~S~q1|q0~T~q1|q0~U~q1|q0~V~q1|q0~W~q1|q0~X~q1|q0~Y~q1|q0~Z~q1|q1~a~q1|q1~b~q1|q1~c~q1|q1~d~q1|q1~e~q1|q1~f~q1|q1~g~q1|q1~h~q1|q1~i~q1|q1~j~q1|q1~k~q1|q1~l~q1|q1~m~q1|q1~n~q1|q1~o~q1|q1~p~q1|q1~q~q1|q1~r~q1|q1~s~q1|q1~t~q1|q1~u~q1|q1~v~q1|q1~w~q1|q1~x~q1|q1~y~q1|q1~z~q1|q1~A~q1|q1~B~q1|q1~C~q1|q1~D~q1|q1~E~q1|q1~F~q1|q1~G~q1|q1~H~q1|q1~I~q1|q1~J~q1|q1~K~q1|q1~L~q1|q1~M~q1|q1~N~q1|q1~O~q1|q1~P~q1|q1~Q~q1|q1~R~q1|q1~S~q1|q1~T~q1|q1~U~q1|q1~V~q1|q1~W~q1|q1~X~q1|q1~Y~q1|q1~Z~q1|q1~0~q1|q1~1~q1|q1~2~q1|q1~3~q1|q1~4~q1|q1~5~q1|q1~6~q1|q1~7~q1|q1~8~q1|q1~9~q1  
4: q0  
5: q1

1. **FA for integer constant**

****

*fa\_int.in*

1: q0|q1|qf|qf0  
2: 0|1|2|3|4|5|6|7|8|9|+|-  
3: q0~0~qf0|q0~+~q1|q0~-~q1|q0~1~qf|q0~2~qf|q0~3~qf|q0~4~qf|q0~5~qf|q0~6~qf|q0~7~qf|q0~8~qf|q0~9~qf|q1~1~qf|q1~2~qf|q1~3~qf|q1~4~qf|q1~5~qf|q1~6~qf|q1~7~qf|q1~8~qf|q1~9~qf|qf~0~qf|qf~1~qf|qf~2~qf|qf~3~qf|qf~4~qf|qf~5~qf|qf~6~qf|qf~7~qf|qf~8~qf|qf~9~qf  
4: q0  
5: qf|qf0

1. **classes**
2. **TransitionInput**
   * auxiliary class for FA class
   * represents the input of the transition function
   * fields
     + state: string
     + symbol: string
3. **TransitionOutput**
   * auxiliary class for FA class
   * represents the output of the transition function
   * fields
     + states: set of string
4. **FA**
   * class implementing a Finite Automaton (FA)
   * fields
     + states: set of string
     + alphabet: set of string
     + transitions: map from TransitionInput instance to TransitionOutput instance
     + initialState: string
     + finalStates: set of string
     + deterministic: boolean
   * constructors
     + FA()
       - Returns a new instance of FA class, ‘fa’, with no parameters.
       - preconditions: true
       - postconditions: ‘fa’ has no parameters
       - return: ‘fa’
     + FA (filePath)
       - Returns a new instance of FA class, ‘fa’, with parameters read from a file.
       - preconditions: ‘filePath’, string – a valid file path
       - postconditions: ‘transitions’, ‘initialState’ and ‘finalStates’ of ‘fa’ are valid (i.e. the states of ‘transitions’, ‘initialState’ and ‘finalStates’ belong to ‘states’; the symbols of ‘transitions’ belong to ‘alphabet’
       - return: ‘fa’
   * other methods and functions
     + isDeterministic()
       - Checks if a FA instance ‘fa’ is deterministic.
       - preconditions: true
       - postconditions: the returned value indicates if ‘fa’ is deterministic
       - return: True, if ‘fa’ is deterministic; False, otherwise
     + isAccepted (sequence)
       - Checks if a sequence ‘sequence’ is accepted by DFA ‘fa’.
       - preconditions: ‘sequence’, string
       - postconditions: the returned value indicates if ‘sequence’ is accepted
       - return: True, if ‘sequence’ is accepted; False, otherwise
       - error: FAException, if ‘fa’ is not deterministic
     + isAcceptedRec(state, sequence)
       - Checks recursively if a sequence ‘sequence’ is accepted by DFA ‘fa’ starting at state ‘state’.
       - preconditions: ‘sequence’, ‘state’ - string
       - postconditions: the returned value indicates if ‘sequence’ is accepted starting at state ‘state’
       - return: True, if ‘sequence’ is accepted; False, otherwise
5. **FAException**
   * exception class
   * extends RuntimeException
6. **FAConsole**
   * class implementing a console for FA class
   * fields
     + fa: FA instance
   * methods and functions
     + start()
       - Runs the FAConsole instance, ‘faConsole’.
       - preconditions: true
       - postconditions: ‘faConsole’ is running
     + readLine()
       - Reads the next line ‘l’ from the console and returns it.
       - preconditions: true
       - postconditions: ‘l’ is the next line read
       - return: ‘l'
     + getCommands()
       - Returns a map ‘m’ from a string to function references of FAConsole instance ‘faConsole’.
       - preconditions: true
       - postconditions: ‘m’ maps to function references of ‘faConsole’
       - return: ‘m'
     + getMenu()
       - Returns a string ‘s’ representing the codes associated with each command description of FAConsole instance ‘faConsole’
       - preconditions: true
       - postconditions: ‘s’ is a string representation of the menu
       - return: ‘s'
     + statesCommand()
       - Writes the states ‘states’ of ‘fa’ to the console.
       - preconditions: true
       - postconditions: ‘states’ were written to the console
     + alphabetCommand()
       - Writes the alphabet ‘alphabet’ of ‘fa’ to the console.
       - preconditions: true
       - postconditions: ‘alphabet’ was written to the console
     + transitionsCommand()
       - Writes the transitions ‘trans’ of ‘fa’ to the console.
       - preconditions: true
       - postconditions: ‘trans’ were written to the console
     + initialStateCommand()
       - Writes the initial state ‘init’ of ‘fa’ to the console.
       - preconditions: true
       - postconditions: ‘init’ was written to the console
     + finalStatesCommand()
       - Writes the final states ‘finals’ of ‘fa’ to the console.
       - preconditions: true
       - postconditions: ‘finals’ were written to the console
     + isAcceptedCommand()
       - Reads a sequence ‘seq’ from the console and writes to the console if it is accepted by the ‘fa’ as long as it is deterministic.
       - preconditions: true
       - postconditions: ‘accepted’ was written to the console if ‘seq’ is accepted by the ‘fa’ or ‘NOT accepted’ otherwise
     + exitCommand()
       - Writes exiting messages ‘em’ to the console.
       - preconditions: true
       - postconditions: ‘em’ were written to the console