КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ
ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

Звіт до лабораторної роботи №3
на тему
«Скінчені автомати.»

Студента 3 курсу ФКНК
групи ТТП-31
Корнієнка Олександра

Київ-2024

Зміст

Вступ

Модуль 3

Розділ 3. Скінчені автомати

Варіант 20

(12 балів). Для заданого натурального k виявити всі слова довжини k, які допускаються скінченим автоматом, мова програмування Haskell

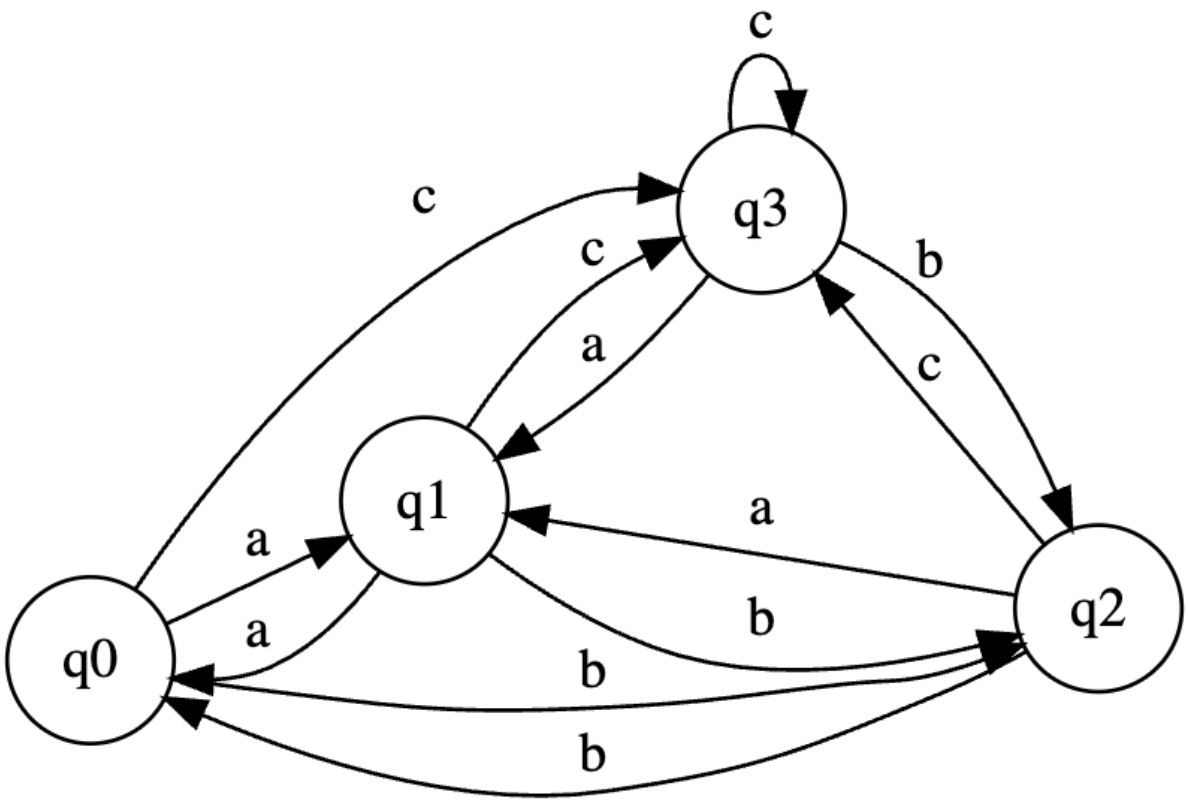
Код програми

```haskell
data Transition = Transition
  { fromState :: String,
    toState :: String,
    symbol :: String
  }

data FiniteAutomaton = FiniteAutomaton
  { states :: [String],
    symbols :: [String],
    transitions :: [Transition],
    startState :: String,
    finalStates :: [String]
  }

generateAllStringsHelper :: FiniteAutomaton -> String -> String -> [String] -> Int ->
[String]
generateAllStringsHelper _ _ currentString allStrings k | length currentString > k =
allStrings
generateAllStringsHelper fa currentState currentString allStrings k
  | length currentString == k && currentState `elem` finalStates fa = currentString :
allStrings
  | otherwise =
      foldr
        ( \transition acc ->
            if fromState transition == currentState
              then generateAllStringsHelper fa (toState transition) (currentString ++ symbol
transition) acc k
              else acc
```

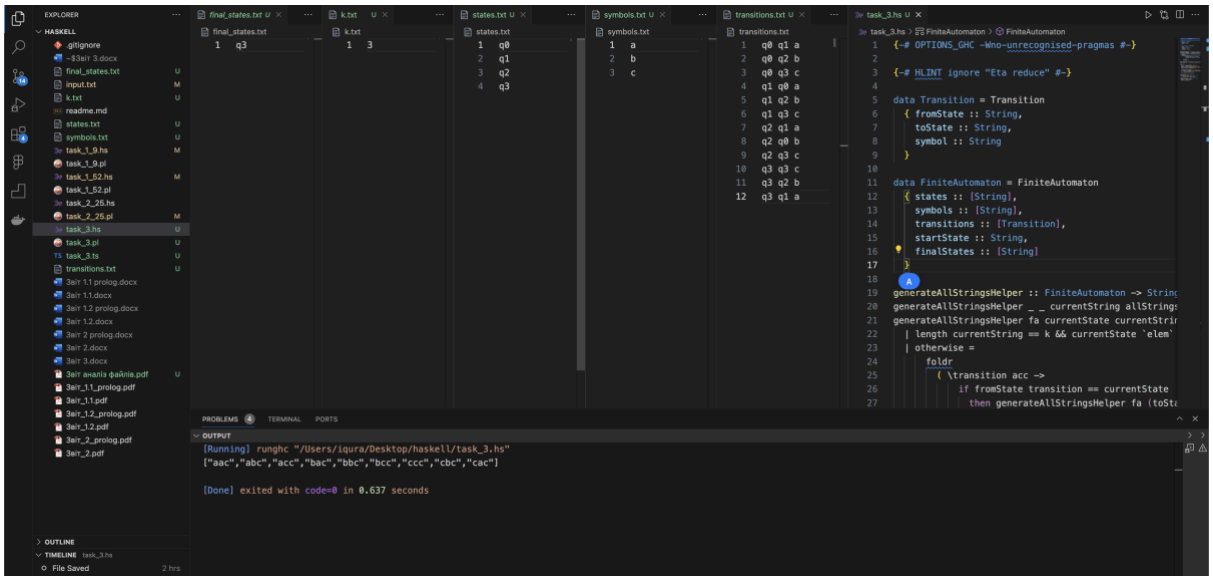
```
        )
        allStrings
        (transitions fa)

generateAllStrings :: FiniteAutomaton -> Int -> [String]
generateAllStrings fa k = generateAllStringsHelper fa (startState fa) "" [] k

main :: IO ()
main = do
  states <- readConstantsFromFile "states.txt"
  symbols <- readConstantsFromFile "symbols.txt"
  finalStates <- readConstantsFromFile "final_states.txt"
  transitionsContent <- readConstantsFromFile "transitions.txt"
  kContent <- readConstantsFromFile "k.txt"

  let startState = head states
  let transitions = map parseTransition transitionsContent
  let automaton = FiniteAutomaton states symbols transitions startState finalStates
  let k = read (head kContent) :: Int

  print $ generateAllStrings automaton k

parseTransition :: String -> Transition
parseTransition line =
  let [fromState, toState, symbol] = words line
   in Transition fromState toState symbol

readConstantsFromFile :: FilePath -> IO [String]
readConstantsFromFile filePath = do
  contents <- readFile filePath
  return $ lines contents
```

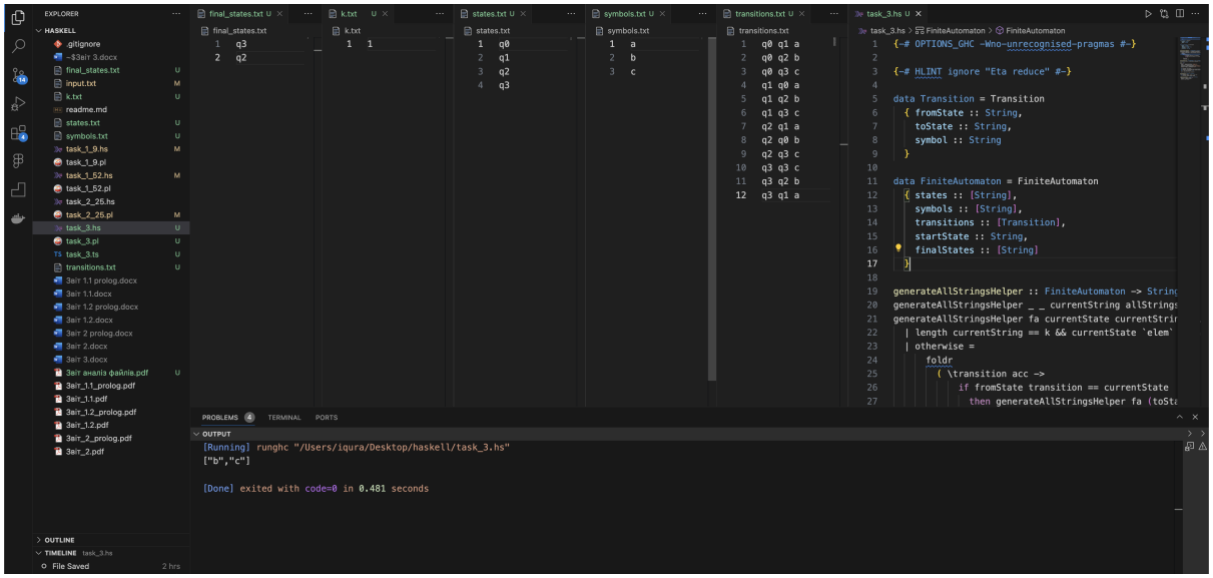
Тести та результати виконання

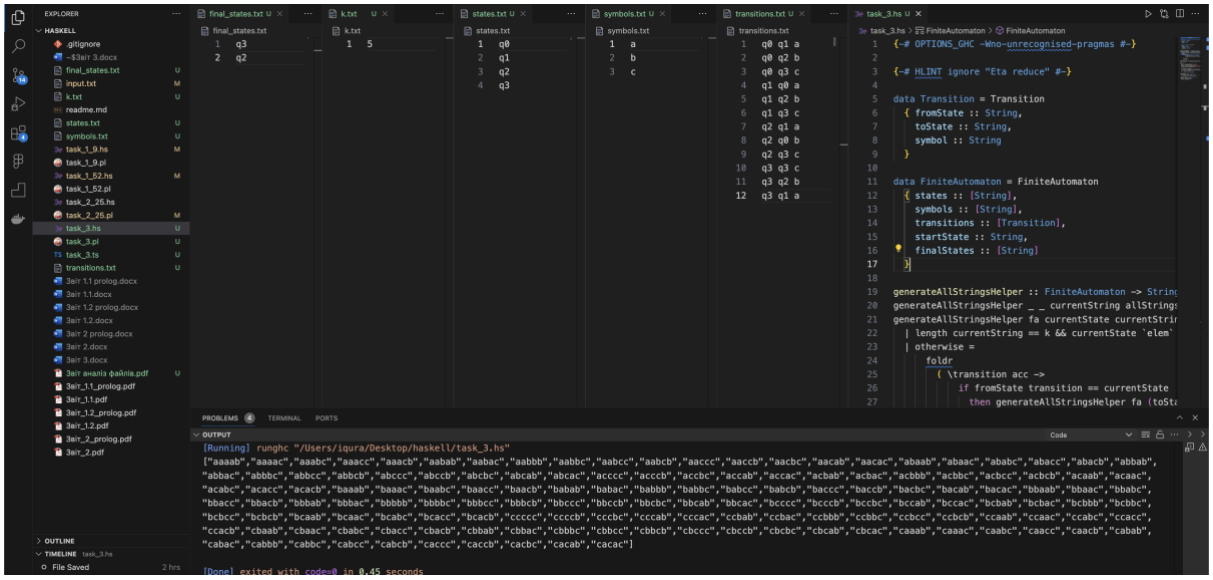
Тести задаються у файлах: final_states.txt, k.txt, states.txt, symbols.txt, transitions.txt

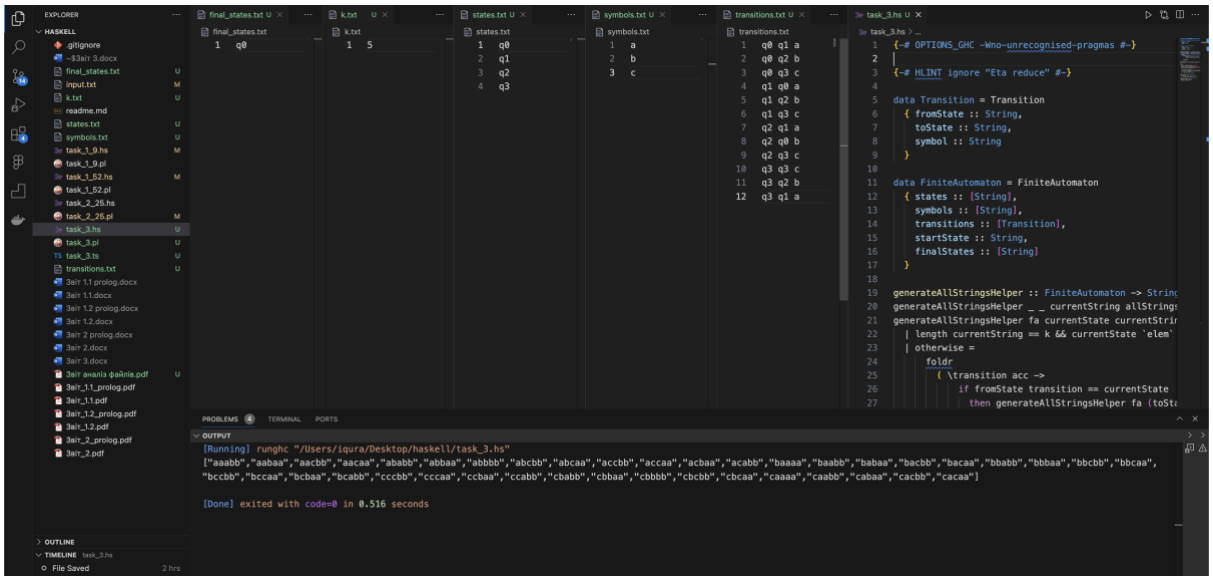


Заданий автомат дозволяє будь-яку послідовність "abc", "bca", "cab", і так далі. Щоб перейти в стан прийняття, останній символ має бути "a" або "b", або "c", або " ". Ось кілька прикладів слів, які можна утворити з цього автомата: abc bca cab acba babc c cc abcc і так далі...
це при умові якщо final state: q0,q1,q2,q3.




final_states.txt
q3
q2
k.txt
1 1
states.txt
q0
q1
q2
q3
symbols.txt
a
b
c
transitions.txt
q0 q1 a
q0 q2 b
q0 q3 c
q1 q0 a
q1 q2 b
q1 q3 c
q2 q1 a
q2 q0 b
q2 q3 c
q3 q3 c
q3 q2 b
q3 q1 a
task_3.hs
{-# OPTIONS_GHC -Wno-unrecognised-pragmas #-}
{-# HLINT ignore "Eta reduce" #-}
data Transition = Transition
{ fromState :: String,
toState :: String,
symbol :: String
}
data FiniteAutomaton = FiniteAutomaton
{ states :: [String],
symbols :: [String],
transitions :: [Transition],
startState :: String,
finalStates :: [String]
}
generateAllStringsHelper :: FiniteAutomaton -> String
generateAllStringsHelper _ _ currentString allStrings
generateAllStringsHelper fa currentState currentStri
| length currentString == k && currentState `elem`
| otherwise =
foldr
(\transition acc ->
if fromState transition == currentState
then generateAllStringsHelper fa (toSt
PROBLEMS 4
TERMINAL
PORTS
OUTPUT
[Running] runghc "/Users/iqura/Desktop/haskell/task_3.hs"
["b","c"]
[Done] exited with code=0 in 0.481 seconds
OUTLINE
TIMELINE task_3.hs
File Saved 2 hrs


final_states.txt
q3
q2
k.txt
1 5
states.txt
q0
q1
q2
q3
symbols.txt
a
b
c
transitions.txt
q0 q1 a
q0 q2 b
q0 q3 c
q1 q0 a
q1 q2 b
q1 q3 c
q2 q1 a
q2 q0 b
q2 q3 c
q3 q3 c
q3 q2 b
q3 q1 a
task_3.hs
{-# OPTIONS_GHC -Wno-unrecognised-pragmas #-}
{-# HLINT ignore "Eta reduce" #-}
data Transition = Transition
{ fromState :: String,
toState :: String,
symbol :: String
}
data FiniteAutomaton = FiniteAutomaton
{ states :: [String],
symbols :: [String],
transitions :: [Transition],
startState :: String,
finalStates :: [String]
}
generateAllStringsHelper :: FiniteAutomaton -> String
generateAllStringsHelper _ _ currentString allStrings
generateAllStringsHelper fa currentState currentStri
| length currentString == k && currentState `elem`
| otherwise =
foldr
(\transition acc ->
if fromState transition == currentState
then generateAllStringsHelper fa (toSt
PROBLEMS 4
TERMINAL
PORTS
OUTPUT
Code
[Running] runghc "/Users/iqura/Desktop/haskell/task_3.hs"
["aaaab","aaaac","aaabc","aaacc","aaacb","aabab","aabac","aabbb","aabbc","aabcc","aabcb","aaccc","aaccb","aacbc","aacab","aacac","abaab","abaac","ababc","abacc","abacb","abbab",
"abbac","abbbc","abbcc","abbcb","abccc","abccb","abcbc","abcab","abcac","acccc","acccb","accbc","accab","accac","acbab","acbac","acbbb","acbbc","acbcc","acbcb","acaab","acaac",
"acabc","acacc","acacb","baaab","baaac","baabc","baacc","baacb","babab","babac","babbb","babbc","babcc","babcb","baccc","baccb","bacbc","bacab","bacac","bbaab","bbaac","bbabc",
"bbacc","bbacb","bbbab","bbbac","bbbbb","bbbbc","bbbcc","bbbcb","bbccc","bbccb","bbcbc","bbcab","bbcac","bccbc","bccbb","bccac","bccab","bcbac","bcbab","bcbbc","bcbbb","bcbbc",
"bcbcc","bcbcb","bcaab","bcaac","bcabc","bcacc","bcacb","ccccc","cccccb","ccccbc","cccab","cccac","ccbab","ccbac","ccbbb","ccbbc","ccbcc","ccbcb","ccaab","ccaac","ccabc","ccacc",
"ccacb","cbaab","cbaac","cbabc","cbacc","cbacb","cbbab","cbbac","cbbbc","cbbcc","cbbcb","cbccc","cbccb","cbcbc","cbcab","cbcac","caaab","caaac","caabc","caacc","caacb","cabab",
"cabac","cabbb","cabbc","cabcc","cabcb","caccc","caccb","cacbc","cacab","cacac"]
[Done] exited with code=0 in 0.45 seconds
OUTLINE
TIMELINE task_3.hs
File Saved 2 hrs


final_states.txt
q0
k.txt
1 5
states.txt
q0
q1
q2
q3
symbols.txt
a
b
c
transitions.txt
q0 q1 a
q0 q2 b
q0 q3 c
q1 q0 a
q1 q2 b
q1 q3 c
q2 q1 a
q2 q0 b
q2 q3 c
q3 q3 c
q3 q2 b
q3 q1 a
task_3.hs
{-# OPTIONS_GHC -Wno-unrecognised-pragmas #-}
{-# HLINT ignore "Eta reduce" #-}
data Transition = Transition
{ fromState :: String,
toState :: String,
symbol :: String
}
data FiniteAutomaton = FiniteAutomaton
{ states :: [String],
symbols :: [String],
transitions :: [Transition],
startState :: String,
finalStates :: [String]
}
generateAllStringsHelper :: FiniteAutomaton -> String
generateAllStringsHelper _ _ currentString allStrings
generateAllStringsHelper fa currentState currentStri
| length currentString == k && currentState `elem`
| otherwise =
foldr
(\transition acc ->
if fromState transition == currentState
then generateAllStringsHelper fa (toSt
PROBLEMS 4
TERMINAL
PORTS
OUTPUT
[Running] runghc "/Users/iqura/Desktop/haskell/task_3.hs"
["aaabb","aabaa","aacbb","aacaa","ababb","abbaa","abbbb","abcbb","abcaa","accbb","accaa","acbaa","acabb","baaaa","baabb","babaa","bacbb","bacaa","bbabb","bbbaa","bbcbb","bbcaa",
"bccbb","bccaa","bcbaa","bcabb","cccbb","cccaa","ccbaa","ccabb","cbabb","cbbaa","cbbbb","cbcbb","cbcaa","caaaa","caabb","cabaa","cacbb","cacaa"]
[Done] exited with code=0 in 0.516 seconds
OUTLINE
TIMELINE task_3.hs
File Saved 2 hrs