51

Mandatory Assignment 1

Regular Languages, Finite Automata, and Regular Expressions

Fariha Nazmul s094747@student.dtu.dk

Abstract

In this assignment we shall assume the existence of a procedure to extract information about the order in which a WHILE program performs the operations *read, write, and nop* (no operation) on a variable.

For this, we assume the alphabet $\sum = \{r, w, n\}$ and the following set of all regular sequences of such operations :

$$E_{\Sigma} = \{e \mid e \text{ is a regular expression over } \Sigma\}$$

Given such a regular expression $e \in E_{\Sigma}$ we want to determine whether all strings in the language $\mathcal{L}(e)$ that e describes have the following property (PROP):

If a *read* occurs then at least one *write* occurred before.

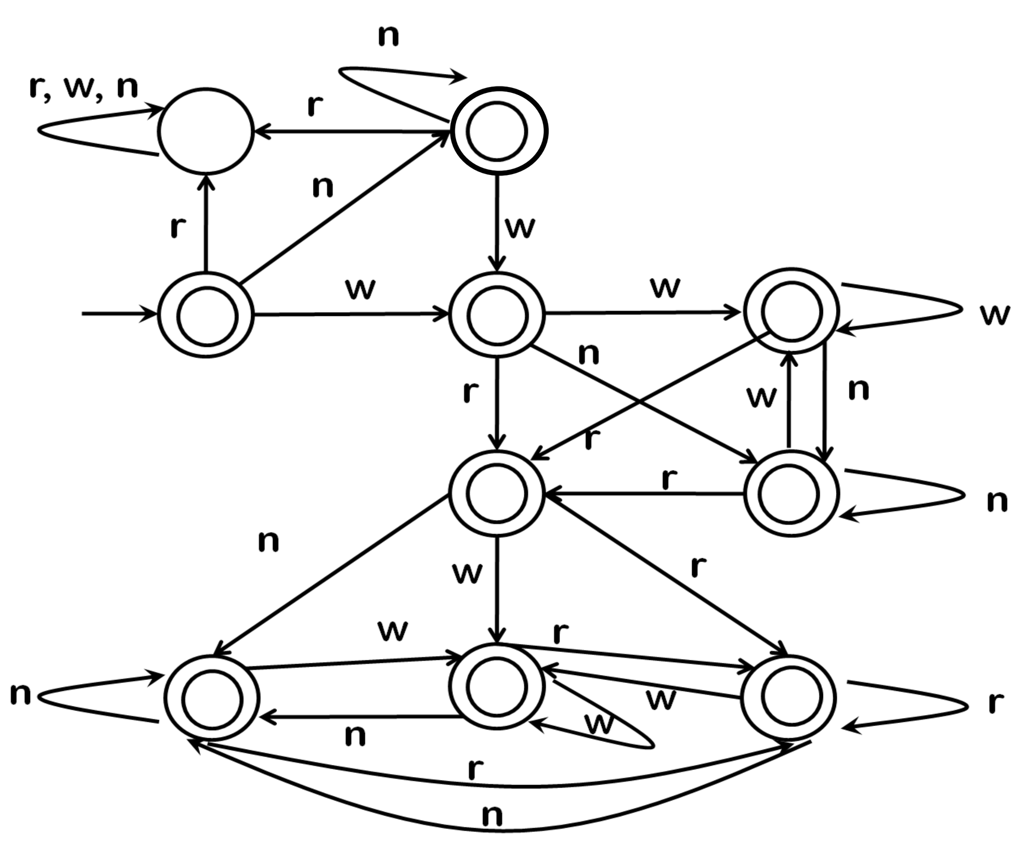
Q1: A Regular Expression Describing L

A regular language describing $\mathcal{L}$ that has the property PROP is:

$$n^* w(n+w)^* r(r+w+n)^* + (n+w)^*$$

Q2: Regular Expression into DFA

The DFA for $\mathcal{L}$ *is*:



Implementation Procedure:

To create the ϵ-NFA from a RegExp I have defined the createNFA() method for all the RegExp.* classes. Then I added the createDFA() method that uses the toDFA() method in FA.Automaton class to convert the ϵ-NFA into DFA using subset construction method.

Q2: Regular Expression into DFA

Implementation Procedure:

The toDFA() method uses the getEClose() method that generates the ϵ-closure of the given state in that automaton and returns the ϵ-closure as a TreeSet<State>.

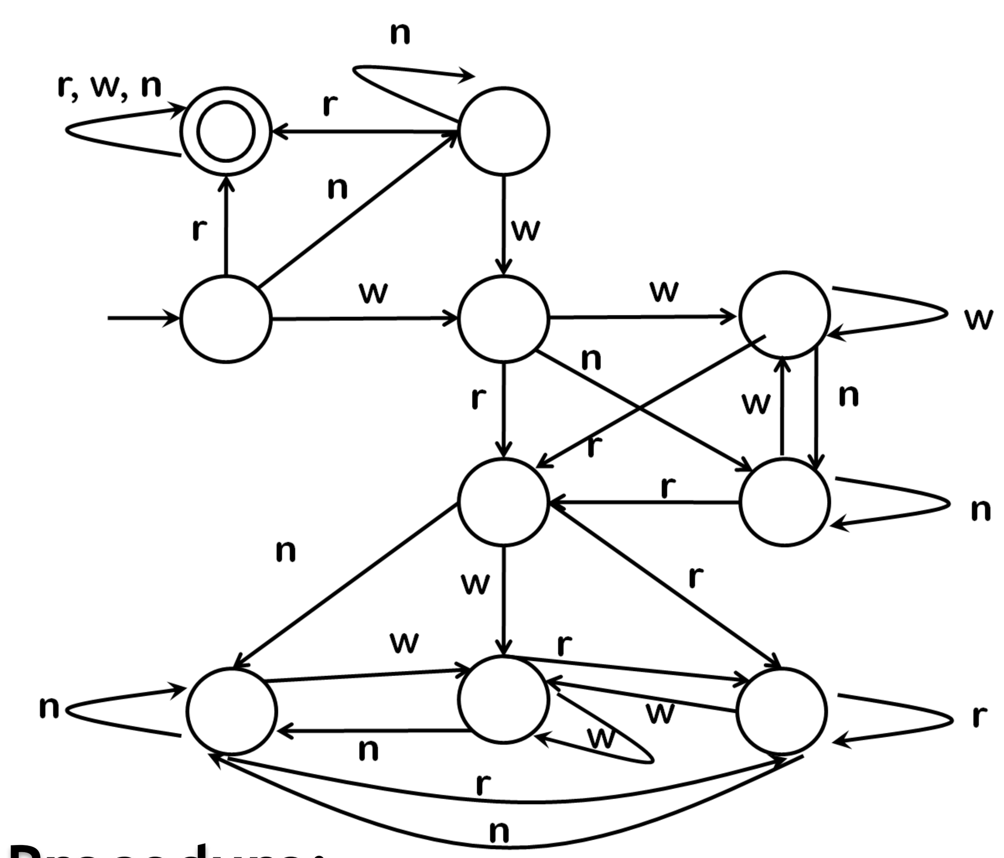
I have used a HashMap for the mapping between the new state of the DFA and its corresponding set of states in the ϵ-NFA .

Algorithm:

- Generate the ϵ-closure of the start state. Add it to the HashMap and set it as the starting state of the DFA.
- Check each element of the set of states for transitions on each alphabet except ϵ and take the union of their ϵ-closure as a state of DFA. Put it in the HashMap if it's a new one.
- Create the transition table of the DFA accordingly.
- Check the transitions for each set of states in the HashMap.
- Mark the final states of the DFA that includes any of the final states of the ϵ-NFA

Q3: DFA for the Complement of Language

The DFA for $\overline{L}$ *is*:



Implementation Procedure:

- Copy all states and transitions of given DFA into a new one.
- In the new DFA set all states as non-accepting.
- Change the non-accepting states of the given DFA into accepting states in the new DFA.

Q4: DFA into Regular Expression

A regular language describing $\overline{L}$:

$$(nn^* r + r)(r + w + n)^*$$

Implementation Procedure:

The RegExp.UnionExp class performs (e_1 + e_2). I have also created a method createUnionDFA() in the FA.Automaton class to create a DFA for L(e_1) U L(e_1).

The toRegExp() method creates the RE from the DFA using the state elimination method. For the RegExps between all states I used the structure
HashMap <State, HashMap<State, RegExp>>.
To trace all the outgoing edges of a state I used
HashMap <State, RegExp>.

Q5: Check a Language for Emptiness

Implementation Procedure:

Check if there is a path from the starting state to final state of the given DFA.

Algorithm:

- Create a set of reachable sets from the starting state of the DFA on transition of any token in the alphabet.
- Check if the set of reachable states contains any final state.

Q6: Check a RE for a Property

Implementation Procedure:

To check the PROP property we need to ensure:

$$L(e) \subseteq L$$

$$\Rightarrow L(e) \cap \neg L = \phi$$

$$\Rightarrow \neg(\neg L(e) \cup L) = \phi$$

Using all the defined methods createDFA(), cmplDFA(), createUnionDFA() and testEmptiness() , the property of a given RegExp e is checked.

Collaborator

Sameer K. C. (s094746)