Union-freeness of Regular Languages

Sukhpal Singh Ghuman
Thapar University
Patiala, Punjab, India

Ajay Kumar
Thapar University
Patiala, Punjab, India

ABSTRACT
A regular language can be converted into an equivalent union-free regular language. Every non-union-free regular expression can be decomposed into an equivalent union-free regular expression, but it may not be unique. In this research paper, an algorithm is designed for determining whether a regular expression is union-free or not and the same is implemented in .NET.

General Terms
Theoretical Computer Science

Keywords
Deterministic finite automata, regular expression, union-free regular language.

1. INTRODUCTION
Regular expressions are well known in the field of computer science. They are commonly used and well-applicable in theory as well as in practice. The regular expressions are used in field of compilers, programming languages, pattern recognition, protocol conformance testing etc..

A regular expression consisting of concatenation, kleene closure and union operations refers to sequential continuation, loops, and branching respectively. In this context, union-free languages represent sequences of operators that do not contain conditional transitions discussed by Sergey Afonin and Denis Golomazov [6]. The union-freeness of languages accelerates the reversal operation on regular language. In this paper an algorithm is designed for checking whether a regular language is union-free or not.

2. RELATED WORK
Sergey Afonin and Denis Golomazov [6] proposed an algorithm for union-free decomposition of regular language. The algorithm uses a set of all maximal finite concatenations of languages. Nagy, B [3] discuss the union-complexity of a regular language. The union-complexity of a language is 1 if and only if it is union-free regular. The decomposing the regular language into an equivalent union-free regular language helps in determining the union complexity of regular language.

3. DEFINITIONS AND NOTATIONS
Regular languages can be described by regular expressions. If r is a regular expression, then the regular language corresponding to r is L(r) discussed by Peter Linz [5]. Union of two regular languages L_1 and L_2 consists of all the strings which are either in L_1 or L_2 discussed by Mishra K.L.P. and N. Chandrasekaran [1].

Def. 1: A regular expression discussed by Ullman [8] over input alphabets Σ can be defined as :

1. Every input alphabet can be represented by itself.
2. Null language and null string also represented by themselves.
3. If r_1 and r_2 are regular expressions representing the languages l_1 and l_2 respectively, then:
   3.1 Union of r_1 and r_2 is represented by $r_1 + r_2$.
   3.2 Kleene closure of the regular expression is represented by $(r_1)^*$.
   3.3 Concatenation of r_1 and r_2 is represented by r_1r_2.
   Rule 3 can be defined recursively.

Example 1: Given regular expression $a+b^*$ denotes the set of all strings {ε, a, b, bb, bbb, bbbb, ...}.

Def. 2: A deterministic finite Automata (DFA) M discussed by Ullman [8] is defined by quintuples $(Q, \Sigma, \delta, q_0, F)$, where Q is the finite non empty set of states, Σ is the finite non empty set of symbols called the input alphabet, δ: $Q \times \Sigma \rightarrow Q$ is called as transition function, q_0 is the initial state, F is subset of Q and set of final states. DFA can be represented by transition diagram or transition table.

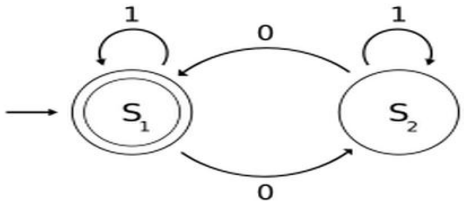

Figure 1: DFA for even number of 0's over {0, 1}

Def. 3: Component of a regular expression is the individual string which is represented by an empty string or by the combination of alphabets, concatenation operator and the star operator as discussed by Sergey Afonin [6].

Example 2: Given a language L represented by regular expression = $\{a + ab^* + a^*b\}$ consists of three components.

Def. 4: Union width of a regular expression is defined as the minimum number of components which are presented in the representation of a regular expression is discussed by Sergey Afonin [6].

Example 3: The regular expression $\{\varepsilon\} + a^*ba^* + b^*ab^*$ has three components and hence the union width is three.

Def. 5: A regular language is said to be union-free if it can be represented by a regular expression, which does not contain the union operation or it is represented by the finite union of union free regular expressions as discussed by Nagy and Sergey Afonin [2,6].

Example 4: The language represented by the regular expression $(a + b^*)^*$, can be converted to an equivalent union-free regular language represented by $(a^*b^*)^*$.

Def. 6: Union-free decomposition of a non-union-free regular language corresponds to the decomposition, which can either be represented by the regular expression which is free from the union operation or the language consist of finite union of union-free regular languages. The resultant language may not be unique as discussed by Nagy and Sergey Afonin [2,6].

Example 5: The regular expression $(a + b)^*$ can be decomposed into $(a^*b^*)^*$ or it can be also be written as $\{\varepsilon\} + a^*ba^* + b^*ab^*$.

4. AN ALGORITHM FOR CHECKING UNION-FREENESS OF A REGULAR LANGUAGE

This algorithm scans the regular expressions from right to left. If right parenthesis occurs immediately after star operator from right to left, counter will become positive. If counter is positive and plus operator appear, then the regular expression is not union-free. If position will become negative one, then regular expression is union-free. The algorithm runs in linear time.

Algorithm 1: Union-freeness of a regular expression(r)

Given a regular expression r, the algorithm determines, whether the language is union-free or not. Position will points to current symbol of regular expression. Initially it points to rightmost symbol of regular expression.

1. Counter =0 and PLUS=0
2. Scan symbol from right to left until leftmost symbol is scanned

```
If Counter=0 then
    If symbol[Position]= star operator
        Position= Position-1
      If (symbol [Position]= right parenthesis)
          Counter= Counter +1
           Position= Position-1
        Endif
    Else
        Position= Position-1
    Endif
Else
    If  symbol[Position]=left parenthesis
       Counter= Counter-1
        Position= Position-1
    Else If (symbol [Position]= right parenthesis)
           Counter= Counter +1
           Position= Position-1
         Else If (symbol [Position]= plus)
                 Print regular expression is not union-free
                  Exit
              Else
                   Position= Position-1
             Endif
         Endif
     Endif
  Endif
```

3. If Position= -1
   Print regular expression is union-free.
   Endif

Example 6: Scan the regular expression $(a(cb)^*+a^*)ab$ from right to left while star operator occurs (4^{th} position from right). Right parenthesis will not occur at 5^{th} position, hence counter remains zero. Next star operator occurs at 7^{th} position from right and right parenthesis occurs at 8^{th} position from right. Counter is set to one, but no plus sign will appear from 9^{th} position from right to leftmost position. Hence it is a union-free regular expression.

5. TOOL FOR ANALYZING WHETHER THE GIVEN REGULAR EXPRESSION IS UNION-FREE OR NOT

The tool for analyzing whether the given regular expression is union-free or not is developed in .NET. This tool will take input a regular expression and determine whether it is union-free or not.

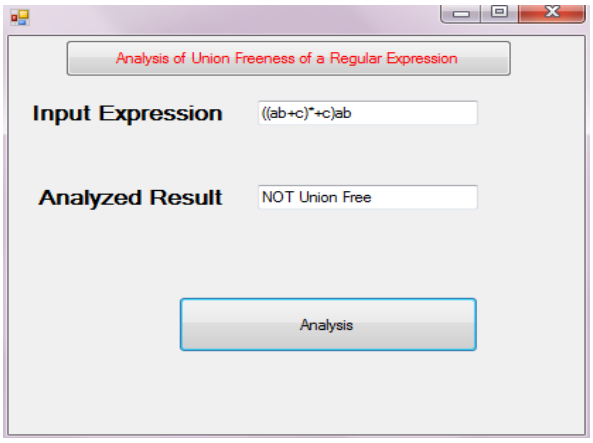

Figure 2: Example of a non-union-free regular expression

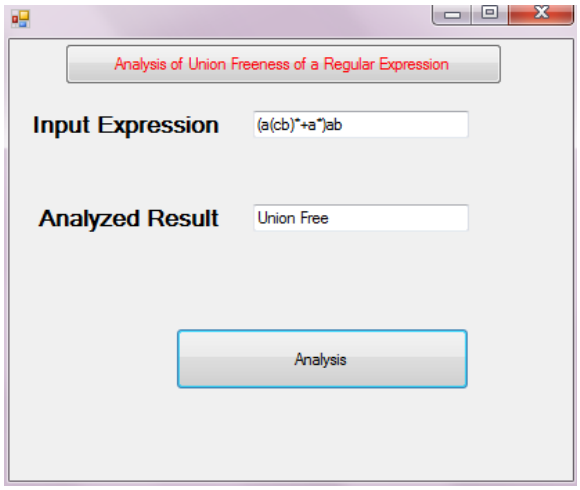

Figure 3: Example of a union-free regular expression

6. CONCUSIONS AND FUTURE WORK

In this paper, we have presented an algorithm which determines whether a given regular expression is union-free or not. A tool is designed using .NET, which will determine whether a regular expression is union-free or not. A regular language which is not union-free can be decomposed into a union-free regular language. In future, an algorithm can be designed to decompose a regular language into an equivalent union-free regular language.

In future, work can be carried out for finding the minimal union-free decomposition a regular language discussed by Sergey Afonin and Denis Golomazov [6]. Software can be designed for the same and can be applied in several applications

7. RFEREENCES

[1] Mishra K.L.P. and N. Chandrasekaran, 1998, " Theory of Computer Science (Automata Language and. Computation) ", PHI, Second edition.

[2] Nagy, B.,2004 , "A normal form for regular expressions ", In Eighth International Conference on Developments in Language Theory, CDMTCS Technical Report 252, Auckland, 51–60.

[3] Nagy, B, 2010, "On Union-complexity of Regular Languages", 11th IEEE International Symposium on Computational Intelligence and Informatics, Hungary, 177-182.

[4] Nagy, B., 2006, "Union-free languages and 1-cycle-free-path-automata", Publicationes mathematicae Debrecen 68, 183– 197.

[5] Peter Linz, 2009, "An Introduction to Formal Languages and Automata", Narosa publishers, fourth edition.

[6] Sergey Afonin and Denis Golomazov, 2009, "On Minimal Union-Free Decompositions of Regular Languages", Third International conference, Lata, Spain, 83-92.

[7] Sinisa Crvenkovic, Igor Dolinka and Zoltan Esik, 2001 "On Equations for Union-Free Regular Languages", Information and Computation, 164, 152-172.

[8] Ullman, J., J. E. Hopcroft and R. Motwani, 2001, "Introduction to Automata Theory, Languages, and Computation", Pearson Education Inc.