IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 3, March 2016.
www.ijiset.com
ISSN 2348 – 7968

About Pumping Lemma for Regular Languages

Boris Tanana[1] and Bhangy Cassy[2]

[1]Higher Institute for Science and Technology of Mozambique (ISCTEM), Maputo, Mozambique

[2]Department of Mathematics and Informatics, Eduardo Mondlane University (UEM), Maputo, Mozambique

Abstract
A powerful technique for showing certain languages not to be regular is the Pumping lemma. In this paper we present one type of problems about Pumping lemma not previously founded in the available scientific literature.
Keywords: *alphabet, word, regular language, deterministic finite automata, transition diagram, pumping lemma.*

1. Introduction

Students in Mozambique, who study the Computer Science Engineering Program, at some subjects they learn some concepts of Discrete Mathematics that are related to regular languages. A powerful technique for showing certain languages not to be regular is the Pumping lemma for regular languages and usually Pumping lemma applies for this kind of problems. See, for example, [1] (pp. 80-88), [2] (pp. 126-130), [3] (pp.212-216). This topic creates many difficulties for our students.

In this paper we present one type of problems about Pumping lemma not previously founded in the available scientific literature. This problems help our students for understanding this topic and the proof of the Pumping lemma.

2. Definitions

Here, at first, we will give the necessary definitions and concepts.

An **alphabet** is a finite, nonempty set whose elements are letters.

A **word** is a finite sequence of letters chosen from some alphabet. The **empty word** is the word with zero occurrences of letters when denotes by λ.
A **length** of the word is the number of positions for letters in the word. The standard notation for the length of a word w is $|w|$.

Let x and y be words. Then xy denotes the **product** (**concatenation**) of x and y, that is, the word formed by making a copy of x and following it by copy of y.

The set of all words on the alphabet $\sum$ is denoted by $\sum^*$. Equipped with product of words it is a monoid, with the empty word as an identity. It is in fact the free monoid on the set $\sum$.

The set of non-empty words is denoted by $\sum^+$. It is the free semigroup on the set $\sum$.

A set L of words, all of which, chosen from some alphabet $\sum$ is called a **language** over $\sum$. That is, L is a subset of $\sum^*$.

The rational operations on languages are the three operations union, product and star, defined as follows:

1) Union $L_1 \cup L_2 = \{u \mid u \in L_1 \; or \mid u \in L_2\}$.
2) Product $L_1 L_2 = \{u_1 u_2 \mid u_1 \in L_1 \; and \mid u_2 \in L_2\}$.
3) Star $L^* = \{u_1 \cdots u_n \mid n \geq 0, \quad and, \quad u_1, \cdots, u_n \in L\}$.

It is also convenient to introduce the operator

$$L^+ = LL^* = \{u_1 \cdots u_n \mid n > 0 \quad and, \quad u_1, \cdots, u_n \in L\}.$$

Note that, L^* is exactly the sub-monoid of $\sum^*$ generated by L, while L^+ is the sub-semigroup of $\sum^*$ generated by L.

The set of **regular** (**rational**) languages over $\sum$ is the smallest set of languages containing the finite languages and closed under finite union, finite product and star.

A **deterministic finite automata** (**DFA**)

$$A = (Q, \sum, \delta, F, q_0)$$

consists of:

Q – a finite set of states;
$\sum$ - an alphabet of input letters;

ISSN 2348 – 7968

δ – a transition function that takes as arguments a state and an input letter and returns a state;
F – a set of final or accepting states;
q_0 – a start state.

Suppose $a_1a_2 \ldots a_n$, is a sequence of input letters, we start out with the DFA in its start state q_0. We consult the transition function δ, say $\delta(q_0, a_1) = q_1$ to find the state that the DFA A enters after processing the first input letter a_1. We process the next input letter, a_2, by evaluating $\delta(q_1, a_2)$; let us suppose this state q_2, we continue in this manner, finding states, $q_3, q_4, \cdots, q_n$, such that, $\delta(q_{i-1}, a_i) = q_i$, for each $i = 1, 2, \cdots, n$.

If, $q_n \in F$, then the input word $w = a_1a_2 \cdots a_n$ is **accepted**, and if not than it is **rejected**. The **language accepted** by DFA A is the set of all words, w that DFA A accepts: $L(A) = \{w \mid w \in \sum^*, \quad A \text{ accepts } w\}$.

A **transition diagram** for a DFA $A = (Q, \sum, \delta, F, q_0)$ is a graph defined as follows:

a) For each state in Q there is a node.
b) For each state q in Q and each letter a in $\sum$, let $\delta(q, a) = p$. Than the transition diagram has an arc from node q to node p, labeled a. If there are several input letters that cause transition from q to p, than the transition diagram can have one arc, labeled by the list of these letters.
c) Nodes corresponding to final states are marked by double circle.
d) There is an arrow into the start state q_0.

Recall here the famous Kleene's Theorem. See, for example, [3] (p. 203).

3. Theorems

Theorem 1. (Kleene): A language L is accepted by DFA A with alphabet of input $\sum$ if, and only if, L is a regular language over $\sum$.

Using the Kleene's Theorem not difficult to proof the Pumping lemma for regular languages. See, for example, [2] (p. 126).

Theorem 2. (The Pumping lemma for regular languages): Let L be a regular language. Then there exists a constant n (which depends on L) such that for every word x in L such that $|x| \geq n$, we can break x into three sub-words, $x = uvw$ such that:

1. $|uv| \leq n$.
2. $|v| \geq 1 (v \neq \lambda)$
3. For all $k \geq 0$, the word $uv^k w$, is also in L.

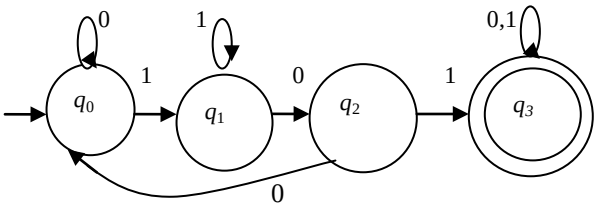
That is, we can always find a nonempty sub-word v of x that can be "pumped"; that is, repeating v any numbers of times, or deleting it (the case k = 0), keeps the resulting word in the language L.

4. Conclusion

As concluding remarks, we, finally present one type of problems about Pumping lemma the promised in the beginning of this article.

Example: Let L be the regular language which denotes by the regular expression, $(0+1)^*101(0+1)^*$. Find the constant n (which depends on L).Choose a word x in L such that $|x| \geq n$. Break x into three sub-words $x = uvw$ such that $|uv| \leq n$, $|v| \geq 1$. Verify that the words, uv , uv^2w is also in L.

Solution: At first we design a DFA A that accept the language L. Here we present the transition diagram of DFA A.



The constant n is equal to the number of states of the DFA A. That is, $n = 4$.

Let $x = 1101$, $|x| \geq 4$ and $x \in L.$; $x = uvw$, where,

$u = 1 \; v = 1 \; w = 01$,
$|uv| = |11| = 2 \leq 4$,
$|v| = |1| = 1 \geq 1$,
$uw = 101 \in L$,
$uv^2w = 11101 \in L$.

ISSN 2348 – 7968

Let choose another word x. For example, $x = 100101$.

$$|x| = 6 \geq 4;\ x \in L;\ x = uvw,$$

where,

$$u = \lambda, v = 100, w = 101,$$
$$|uv| = |100| = 3 \leq 4,$$
$$|v| = |100| = 3 \geq 1,$$
$$uw = 101 \in L,$$
$$uv^2 w = 100100101 \in L.$$

The problems of this type to validate actual student's understanding of Pumping lemma and there proof.

References

[1] Ding – Zhu Du, Ker – I Ko. Problem Solving in Automata, Languages, and Complexity, John Wiley & Sons, Inc, 2001
[2] Jonh E. Hopcroft, Rajeev Motwani, Jeffery D. Uiiman. Introduction to Automata Theory, Languages, and Computation, (2nd ed.), Addison – Wesley, 2001
[3] Thomas A. Sudkamp. Languages and Machines, An Introduction to the Theory of Computer Science (2nd. ed), Addison – Wesley, 1998

Boris Tanana. Achieved his PhD degree in 1980. He was employed at Ural's State University (URGU) in Russia, then at Eduardo Mondlane University and Higher Institute for Science and Technology, both, in Maputo, Mozambique as Associate Professor. He has more than 50 scientific papers published. His areas of research interest, topological semigroups, regular languages and pseudo varieties of finite monoids.

Bhangy Cassy. PhD in Mathematics Education. Full Professor at Eduardo Mondlane University in Maputo Mozambique. The scientific interest areas are related to Algebra, Mathematical Logic, Mathematics Education, History and Philosophy of Mathematics.