Pumping Lemma for Non-Regular Languages

Chih Chin Chang

Department of Information and Computer Science

University of Hawaii at Manoa

Honolulu, Hawaii

changcc@hawaii.edu

Abstract

There are many classifications of a hierarchy of languages, and one of them is the class of regular languages. Regular languages are limited to those that can be expressed by regular expressions. This paper studies a particular property of regular languages called the pumping lemma, which is used to disprove that a certain language is not regular. In particular, for a language specified by the alphabets including 0 and 1 such that the greatest common divisor of the number of zeros and the number of ones is 1, the pumping lemma can be employed to demonstrate that such a language is not regular.

Keywords

pumping lemma; regular language; greatest common divisor; prime numbers;

I. Introduction

Generally, formal languages can be defined rigorously. A rigorously defined language, usually denoted by the capital letter L, is a set of strings over an alphabet, Σ, which is a set of symbols. Strings can be empty, individual alphabet symbols, or constructed by concatenation of symbols in an alphabet. And the set of all possible strings given an alphabet is the Kleene Closure of the alphabet, Σ^*.

Languages can be defined using the set builder notation and classified in a hierarchy of complexity. One class of languages in Chomsky's Hierarchy is the regular languages. These languages can be expressed by regular expressions and accepted by finite state automata.

A. Prime Numbers

Prime numbers are positive integers whose factors are only 1 and the number itself. More formally, for any prime number p, there does not exist any positive integer t, such that $1<t<p$, where $p\%t = 0$.

Two positive integers are relatively prime if their greatest common divisor is 1.

B. Greatest Common Divisor

The greatest common divisor is a function denoted by gcd(), taking two inputs, a and b, from the set $\mathbb{N}$ to determine the largest positive integer, k, such that $a\%k$ = 0 and $b\%k$ = 0.

Intuitively, gcd is a function that searches through the factors of two positive integers and returns the max integer that is common to both numbers.

C. A particular language L

For the purposes of this paper, a particular language, L, is analyzed. Specifically, let $\Sigma = \{0, 1\}$ and L over Σ = $\{0^i 1^j | gcd(i, j) = 1\}$, and the subsequent sections will prove that this language is not regular.

II. Proof

A. Relationship between GCD and Prime Numbers

In the universal domain of natural numbers, $\mathbb{N}$, any non-prime numbers can be factored down into a multiplication of only prime numbers by the definition of prime numbers. Such that suppose P denotes the set of all prime numbers in $\mathbb{N}$, and Q denotes the set difference, $\mathbb{N}$ - P, then for any arbitrary number, q, $\in$ Q, there exists a string of prime numbers, p $\subseteq \Sigma^*$, such that when every number in p is multiplied would equal q. This is a recursive definition that can be justified by considering the following cases:

For any integer w $\in \mathbb{N}$
case 1: w is prime
If w is a prime integer, then it can be factored by multiplying 1 and w, which are both prime numbers.
case 2: w is not prime
By the definition of prime numbers, there exists at least one positive integer, k, such that $1<k<w$. Where $w\%k = 0$.

Considering any arbitrary k, In the case that k is prime, it is trivial that the factors of w are prime. However, in the case where k is not a prime, then the same case 2 applies to the number k until the factor is again prime.

Such is the relationship between the function gcd and prime numbers, that any positive integer can be reduced to multiples of prime factors. And therefore, the common factors of any two numbers must include prime numbers. The subsequent sections will use this property to deduce that gcd(i,j) may be greater than 1, keeping in mind that gcd(i, j) = 1 suggests that i and j are relatively prime by definition.

B. Relationship between Pumping Lemma and Regular Languages

Regular languages can be expressed by regular expressions and accepted by finite state automata. For languages that are finite, meaning that there is a finite number of strings accepted by this language, it is trivial for the construction of a finite state automata and regular expression. Simply create states for every acceptable strings. However, for an inifinite language, like the given problem L, the machine must contain loops. That is to say that there exist a positive integer, n, such that a string within the language L whose length exceeds n will require the machine to loop. At the same time, a string accepted by the machine M that includes looping can be pumped down, reducing the number of loops, without its acceptance into L being changed. This is the introduction to the pumping lemma, which intuitively, describes that for an infinite language, the machine needs some sort of loop to pump strings whose length is greater than n.

This suggests the limits of the FSA machine. That assuming a FSA that accepts a language L, and if the pumping of the machine will produce a string not accepted by the language, then the contradiction must be true: that the machine does not exist, and therefore, the langauge is not regular.

This proof must follow a certain procedure. Consider a language L, select a string z in L whose length may depend on n. The adversary, who is any arbitrary being, breaks z into the concatenation of u, v, and w so that $|uv| \leq n$ and $|v| \geq 1$. This step suggests that v is the loop within the machine. Therefore, pumping the loop up or down should not change the string's acceptance into L. And lastly, construct a contradiction by manipulating i such that uv^iw is not within the language. If such a construction is successful, then the language is not regular.

C. Proof on the particular language L

Consider the current given language, L = $\{0^i1^j|gcd(i,j) = 1\}$, the pumping lemma and the previously described gcd property will be used to prove that L is not regular.

Assume a FSA machine M with n states that accepts L(G) so that L is assumed to be regular. It is obvious that L is not finite since there are an infinite combinations of i and j that are relatively prime.

Choose z = 0^a1^b such that b is constructed by the multiples of primes up to the nth prime number, or intuitively, b = 2*3*5*7*...*p_n where p_n is the nth prime number. And a = p_{n+1}. This forces that $|0|>n$. Also since a is a prime number, the gcd(a, b) = 1 because, by the definition of prime numbers, since a is prime, it has no other factors other than 1 and itself, which is not common with any factors of b. And because gcd(a, b) = 1, z is in L.

The adversary must break z into uvw where $|uv| \leq n$ and $|v| \geq 1$. Since a$>n$, it follows that u = $0^{|u|}$ and v = $0^{|v|}$.

Choose i = 0, if $|u|$>1, and i = 2 otherwise, so that v is pumped down.

- case a: $|u|$=0

  Then $0^{|v|}1^b$=0^21^b. And because 2 is included in the construction of b as a multiple of primes, gcd(2, b)>1, and therefore the string is not included in L.
- case b: $|u|$=1

  Then $0^{1+2}1^b$. And because 3 is included in the construction of b, gcd(3, b)>1, and therefore the string is not included in L.
- case c: for all other values of $|u|$

  The string uv^iw = uv^0w = uw = $0^{a-|v|}1^b$. Because $|v| \geq 1$, the following cases apply:

  Let q = $a - |v|$

  - case 1: q is a prime

    Because b is constructed by the multiples of every prime leading up to , but not including, a, if q is a prime, then it matches one of the prime number in b. Therefore, the gcd(q, b) = q, which is greater than 1, and thus not acceptable in the language L.

- case 2: q is not a prime

  By the relationship defined between gcd and prime numbers, if q is not a prime, then it can be factored into multiples of prime numbers. And since b is constructed by the multiples of every prime leading up to, but not including, a, the factors of q must be included in the primes of b. Therefore, $gcd(q, b) > 1$.

III. Conclusion

In conclusion, the language defined L = $\{0^i 1^j | gcd(i, j) = 1\}$ is not regular because the pumping lemma can produce a string z' that is not within the language from a string z that is within the language. This paper demonstrates the contradiction argument to prove that a language cannot be rigorously accepted by a finite state automata and therefore cannot be a regular language. At the same time, this paper used fundamental definitions of prime numbers and the function greatest common divisor to uncover a new property such that every positive integers can be broken down into multiples of prime numbers. And using this property, the pumping demonstration proves that the language is not regular.

References

[1] W. S. Brainard and L. H. Landweber, *Theory of Computation*, John Wiley & Sons, 1974.

[2] J. Carroll and D. Long, *Theory of Finite Automata with an Introduction to Formal Languages*, Englewood Cliffs, New Jersey: Prentice-Hall, 1989.

[3] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[4] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation* Addison-Wesley, 1979.

[5] K. H. Rosen, *Discrete Mathematics and Its Applications*, 6th ed. New York, United States: McGraw-Hill, 2007.