6. Regular and Non-Regular Languages

Theorem: There is a countably infinite number of regular languages.

Proof: We can lexicographically enumerate all the syntactically legal DFSMs with input alphabet ∑. Every regular language is accepted by at least one of them. So there cannot be more regular languages than there are DFSMs. Thus there are at most a countably infinite number of regular languages. There is not a one-to-one relationship between regular languages and DFSMs since there is an infinite number of machines that accept any given language. But the number of regular languages is infinite because it includes the following infinite set of languages:

{a}, {aa} , {aaa}, {aaaa}. {aaaaa}, {aaaaaa }…

Theorem: Every finite language is regular.

Proof: If L is the empty set, then it is defined by the regular expression Φ and so is regular. If it is any finite language composed of the strings s_1, s_2, ... s_n for some positive integer n, then it is defined by the regular expression: $s_1 \cup s_2 \cup \cdots \cup s_n$

So it too is regular.

Example: The Intersection of Two Infinite Languages

Let L = $L_1 \cap L_2$, where L_1 = { $a^n b^n$; n≥0} and L_2 ={ $b^n a^n$; n≥0} As we will soon be able to prove, neither L_1 nor L_2 is regular. But L is. L = {e}, which is finite.

Example: A Finite Language We May Not Be Able to Write Down

Let L = {w ε {0 - 9}* : w is the social security number of a living US resident}.

L is regular because it is finite. It doesn't matter that no individual or organization happens at any given instant, to know what strings are in L.

Note: Not every regular language is computationally tractable(manageable). Example is Towers of Hanoi language.

But, of course, most interesting regular languages are infinite. Sn far. we've developed four techniques for showing that a (finite or infinite) language L is regular:

- Exhibit a regular expression for L.
- Exhibit an FSM for L.
- Show that the number of equivalence classes of $\approx_L$ is finite.
- Exhibit a regular grammar for L

6.1 Closure Properties of Regular Languages

Theorem: The regular languages are closed under union, concatenation, and Kleene star.

Proof: By the same constructions that were used in the proof of Kleene's theorem

Theorem: The regular languages are closed under complement, intersection, difference, reverse, and letter substitution.

Proof:

- The regular languages are closed under complement. If L_1 is regular, then there exists a DFSM $M_1 = (K, \Sigma, \delta, s, A)$ that accepts it. The DFSM $M_2 = (K, \Sigma, \delta, s, K - A)$, namely M_1 with accepting and nonaccepting states swapped, accepts $\neg(L(M_1))$ because it rejects all strings that M_1 accepts and rejects all strings that M_1 accepts.
- The regular languages are closed under intersection. We note that:

$$L(M_1) \cap L(M_2) = \neg(\neg L(M_1) \cup \neg L(M_2)).$$

We have already shown that the regular languages are closed under both complement and union. Thus they are also closed under intersection.

- The regular languages are closed under set difference (subtraction). We note that:

$$L(M_1) - L(M_2) = L(M_1) \cap \neg L(M_2).$$

We have already shown that the regular languages are closed under both complement and intersection. Thus they are also closed under set difference.

- The regular languages are closed under reverse. Recall that $L^R = \{w \in \Sigma^* : w = x^R \text{ for some } x \in L\}$. We leave the proof of this as an exercise.
- The regular languages are closed under letter substitution, defined as follows: Consider any two alphabets, Σ_1 and Σ_2. Let *sub* be any function from Σ_1 to Σ_2^*. Then *letsub* is a letter substitution function from L_1 to L_2 iff $letsub(L_1) = \{w \in \Sigma_2^* : \exists y \in L_1 (w = y \text{ except that every character } c \text{ of } y \text{ has been replaced by } sub(c))\}$. For example, suppose that $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{0, 1\}$, $sub(a) = 0$, and $sub(b) = 11$. Then $letsub(\{a^n b^n : n \geq 0\}) = \{0^n 1^{2n} : n \geq 0\}$. We leave the proof that the regular languages are closed under letter substitution as an exercise.

6.2 Showing That a Language is Not Regular

The Pumping Theorem for Regular languages

Theorem: If L is a regular language, then:

$$\exists k \geq 1 \ (\forall \text{ strings } w \in L, \text{ where } |w| \geq k \ (\exists x, y, z \ (w = xyz, |xy| \leq k, y \neq \varepsilon, \text{ and } \forall q \geq 0 \ (xy^q z \in L))))$$

Proof: The proof is the argument that we gave above: If L is regular then it is accepted by some DFSM $M = (K, \Sigma, \delta, s, A)$. Let k be $|K|$. Let w be any string in L of length k or greater. By Theorem 8.5, to accept w, M must traverse some loop at least once. We can carve w up and assign the name y to the first substring to drive M through a loop. Then x is the part of w that precedes y and z is the part of w that follows y. We show that each of the last three conditions must then hold:

- $|xy| \leq k$: M must not only traverse a loop eventually when reading w, it must do so for the first time by at least the time it has read k characters. It can read $k - 1$ characters without revisiting any states. But the k^{th} character must, if no earlier character already has, take M to a state it has visited before. Whatever character does that is the last in one pass through some loop.
- $y \neq \varepsilon$: Since M is deterministic, there are no loops that can be traversed by ε.
- $\forall q \geq 0 \ (xy^q z \in L)$: y can be pumped out once (which is what happens if $q = 0$) or in any number of times (which happens if q is greater than 1) and the resulting string must be in L since it will be accepted by M. It is possible that we could chop y out more than once and still generate a string in L, but without knowing how much longer w is than k, we don't know any more than that it can be pumped out once.

In a nutshell then, to use the Pumping Theorem to show that a language L is not regular, we must:

1. Choose a string w, where $w \in L$ and $|w| \geq k$. Note that we do not know what k is; we know only that it exists. So we must state w in terms of k.
2. Divide the possibilities for y into a set of equivalence classes so that all strings in a class can be considered together.
3. For each such class of possible y values, where $|xy| \leq k$ and $y \neq \varepsilon$:
   Choose a value for q such that $xy^q z$ is not in L.

a^nb^n is not regular

Let L be $A^nB^n = \{a^nb^n : n \geq 0\}$. We can use the Pumping Theorem to show that L is not regular. If it were, then there would exist some k such that any string w, where $|w| \geq k$, must satisfy the conditions of the theorem. We show one string w that does not. Let $w = a^kb^k$. Since $|w| = 2k$, w is long enough and it is in L, so it must satisfy the conditions of the Pumping Theorem. So there must exist x, y, and z, such that $w = xyz$, $|xy| \leq k$, $y \neq \varepsilon$, and $\forall q \geq 0\ (xy^qz \in L)$. But we show that no such x, y, and z exist. Since we must guarantee that $|xy| \leq k$, y must occur within the first k characters and so $y = a^p$ for some p. Since we must guarantee that $y \neq \varepsilon$, p must be greater than 0. Let $q = 2$. (In other words, we pump in one extra copy of y.) The resulting string is $a^{k+p}b^k$. The last condition of the Pumping Theorem states that this string must be in L, but it is not since it has more a's than b's. Thus there exists at least one long string in L that fails to satisfy the conditions of the Pumping Theorem. So $L = A^nB^n$ is not regular.

The Balanced Parenthesis Language is Not Regular

Let L be Bal $= \{w \in \{), (\}^* :$ the parentheses are balanced$\}$. If L were regular, then there would exist some k such that any string w, where $|w| \geq k$, must satisfy the conditions of the theorem. Bal contains complex strings like (())((()()). But it is almost always easier to use the Pumping Theorem if we pick as simple a string as possible. So, let $w = (^k)^k$. Since $|w| = 2k$ and w is in L, w must satisfy the conditions of the Pumping Theorem. So there must exist x, y, and z, such that $w = xyz$, $|xy| \leq k$, $y \neq \varepsilon$, and $\forall q \geq 0\ (xy^qz \in L)$. But we show that no x, y, and z exist. Since $|xy| \leq k$, y must occur within the first k characters and so $y = (^p$ for some p). Since $y \neq \varepsilon$, p must be greater than 0. Let $q = 2$. (In other words, we pump in one extra copy of y.) The resulting string is $(^{k+p})^k$. The last condition of the Pumping Theorem states that this string must be in L, but it is not since it has more ('s than)'s. There exists at least one long string in L that fails to satisfy the conditions of the Pumping Theorem. So $L =$ Bal is not regular.

The Even Palindrome language is Not Regular

Let L be PalEven $= \{ww^R : w \in \{a, b\}^*\}$. PalEven is the language of even-length palindromes of a's and b's. We can use the Pumping Theorem to show that PalEven is not regular. If it were, then there would exist some k such that any string w, where $|w| \geq k$, must satisfy the conditions of the theorem. We show one string w that does not. (Note here that the variable w used in the definition of L is different from the variable w mentioned in the Pumping Theorem.) We will choose w so that we only have to consider one case for where y could fall. Let $w = a^kb^kb^ka^k$. Since $|w| = 4k$ and w is in L, w must satisfy the conditions of the Pumping Theorem. So there must exist x, y, and z, such that $w = xyz$, $|xy| \leq k$, $y \neq \varepsilon$, and $\forall q \geq 0\ (xy^qz \in L)$. Since $|xy| \leq k$, y must occur within the first k characters and so $y = a^p$ for some p. Since $y \neq \varepsilon$, p must be greater than 0. Let $q = 2$. The resulting string is $a^{k+p}b^kb^ka^k$. If p is odd, then this string is not in PalEven because all strings in PalEven have even length. If p is even then it is at least 2, so the first half of the string has more a's than the second half does, so it is not in PalEven. So $L =$ PalEven is not regular.

The Language with More a's than b's is Not Regular

Let $L = \{a^n b^m : n > m\}$. We can use the Pumping Theorem to show that L is not regular. If it were, then there would exist some k such that any string w, where $|w| \geq k$, must satisfy the conditions of the theorem. We show one string w that does not. Let $w = a^{k+1}b^k$. Since $|w| = 2k + 1$ and w is in L, w must satisfy the conditions of the Pumping Theorem. So there must exist x, y, and z, such that $w = xyz$, $|xy| \leq k$, $y \neq \varepsilon$, and $\forall q \geq 0\ (xy^q z \in L)$. Since $|xy| \leq k$, y must occur within the first k characters and so $y = a^p$ for some p. Since $y \neq \varepsilon$, p must be greater than 0. There are already more a's than b's, as required by the definition of L. If we pump in, there will be even more a's and the resulting string will still be in L. But we can set q to 0 (and so pump out). The resulting string is then $a^{k+1-p}b^k$. Since $p > 0$, $k + 1 - p \leq k$, so the resulting string no longer has more a's than b's and so is not in L. There exists at least one long string in L that fails to satisfy the conditions of the Pumping Theorem. So L is not regular.

The Prime Number of a's Language is Not Regular

Let L be $\text{Prime}_a = \{a^n : n \text{ is prime}\}$. We can use the Pumping Theorem to show that L is not regular. If it were, then there would exist some k such that any string w, where $|w| \geq k$, must satisfy the conditions of the theorem. We show one string w that does not. Let $w = a^j$, where j is the smallest prime number greater than $k + 1$. Since $|w| > k$, w must satisfy the conditions of the Pumping Theorem. So there must exist x, y, and z, such that $w = xyz$, $|xy| \leq k$ and $y \neq \varepsilon$. $y = a^p$ for some p. The Pumping Theorem further requires that $\forall q \geq 0\ (xy^q z \in L)$. So, $\forall q \geq 0$ ($a^{|x|+|z|+q|y|}$ must be in L). That means that $|x| + |z| + q \cdot |y|$ must be prime.

But suppose that $q = |x| + |z|$. Then:

$$\begin{aligned} |x| + |z| + q \cdot |y| &= |x| + |z| + (|x| + |z|) \cdot y \\ &= (|x| + |z|) \cdot (1 + |y|), \end{aligned}$$

which is composite (non-prime) if both factors are greater than 1. $(|x| + |z|) > 1$ because $|w| > k + 1$, and $|y| \leq k$. $(1 + |y|) > 1$ because $|y| > 0$. So, for at least that one value of q, the resulting string is not in L. So L is not regular.

When we do a Pumping theorem proof that a language L is not regular, we have two choices to make: a value for w and a value for q. As we have just seen, there are some useful heuristics that can guide our choices:

To choose *w:*

- Choose a w that is in the part of L that makes it not regular.
- Choose a w that is only barely in L.
- Choose a w with as homogeneous as possible an initial region of length at least k.

To choose *q:*

- Try letting q be either 0 or 2.
- If that doesn't work, analyze L to see if there is some other specific value that will work.

*****