1 Showing Languages are Non-Regular

Question: How can one show that a language is not regular?

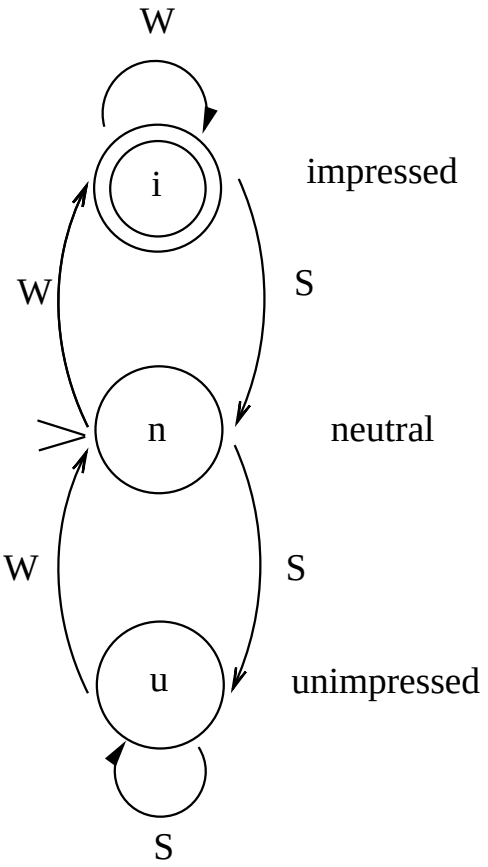
- We have no way to do this so far; constructing a finite automaton or a regular expression can only show a language is regular.
- To show a language is not regular, one would have to consider all possible finite automata or regular expressions.

It could be helpful to show a language is non-regular to avoid wasting time looking for a finite automaton or a regular expression for it.

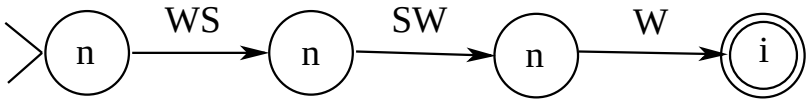
- One would then know that a finite amount of memory is not enough to recognize the language.
- This shows that more powerful techniques such as context-free languages or Turing machines are needed for the language.
- This material will help you to have intuition which kinds of languages can be described by finite automata and which cannot.
- This will help you to see, for example, that programming languages cannot be described by finite automata, in most cases.

To show that a language is non-regular, we have to find some property P that all regular languages have, and then to show that a language is not regular, we have to show that it does not have property P.

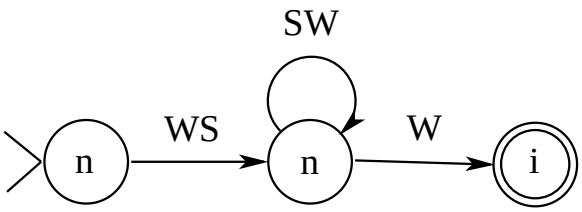
To obtain such a property P, consider again the interview automaton:



Consider the input $WSSWW$. The prefixes WS and $WSSW$ both lead to the state n.



Thus the substring SW of $WSSWW$ leads from n to n:



This means that $WS(SW)^2W$, that is, $WSSWSWW$, is also accepted, and WSW is accepted, and in fact, $WS(SW)^iW$ is accepted for any i.

What has to be true of an input to be able to pump it like this?

Note that the string W by itself is accepted, but it cannot be pumped by the same argument. For this automaton, such pumping can be done on any string of length three or more that is accepted. Why is this?

In general, for a finite automaton with n states, the same argument can be done for any string of length greater than or equal to n.

Theorem 1.1 (2.4.1) *Let L be a regular language. Then*

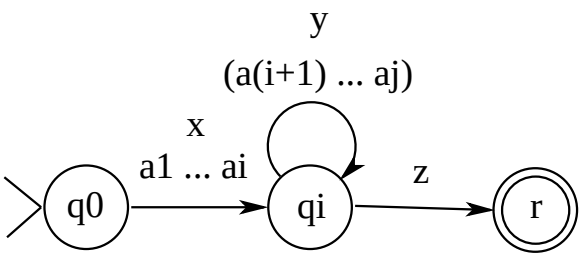
- *there is an integer $n \geq 1$ such that*
- *any string $w \in L$ with $|w| \geq n$ can be written as $w = xyz$ where x, y, and z are strings and $y \neq \epsilon$, $|xy| \leq n$ and*
- *$x(y^k)z \in L$ for all $k \geq 0$.*

Proof: Because L is regular, there is a finite automaton M such that L is the language recognized by M.

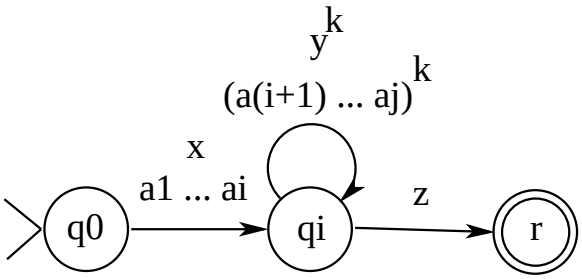
- Let n be the number of states of M.
- Let $a_1, a_2, \ldots, a_n$ be the first n symbols of w. Let q_0 be the start state of M, and let q_i be the state M is in after reading the symbols $a_1, a_2, \ldots, a_i$.
- The sequence $q_0, q_1, q_2, \ldots, q_n$ of states has $n + 1$ elements, but there are only n states in M.
- Thus there must be i and j with $i \neq j$ such that $q_i = q_j$.
- Let x be $a_1, a_2, \ldots, a_i$, let y be $a_{i+1}, a_{i+2}, \ldots, a_j$, and let z be the rest of w, that is, $xyz = w$.
- Then x causes M to go from q_0 to q_i, y causes M to go from q_i to q_j, which is equal to q_i, and z causes M to go from q_j to some accepting state r of M.
- Then y^k causes M to go from q_i to q_i for any k.
- Thus for any k, the string $x(y^k)z$ is accepted, because x causes M to go from q_0 to q_i, y^i causes M to go from q_i to q_i, and z causes M to go from q_i to the accepting state r of M.

- Therefore the string $x(y^k)z$ is in L.

Diagram of the computation for the string $w = xyz$:



The computation for the "pumped" string xy^kz:



Corollary 1.1 *A language L is not regular if,*

- *for all integers $N > 0$,*
- *there exists a string w in L with $|w| \geq N$, such that*
- *for all ways of expressing w as xyz with $|xy| \leq N$, and $y \neq \epsilon$,*
- *there exists a $k \geq 0$ such that $x(y^k)z \notin L$.*

1.1 Showing $a^n b^n$ non-regular

We illustrate this result on the language $L = \{a^n b^n : n \geq 0\}$.

To show the property of the corollary, it is necessary to consider all integers N. This is an infinite number of cases, so the work has to be made finite by mathematics.

To show that L is not regular, first, for all $N > 0$ it is necessary to choose a string w in L with $|w| \geq N$.

- For this, we choose the string $a^N b^N$.
- Then, we have to show that for all ways of expressing $a^N b^N$ as xyz with $|xy| \leq N$, and $y \neq \epsilon$, there exists a $k \geq 0$ such that $x(y^k)z \notin L$.

For this, we consider two possibilities:

1. Suppose y has unequal numbers of a and b.

   - Then $x(y^2)z \notin L$ because $x(y^2)z$ has as many a's and b's as xyz, plus the a's and b's in y.
   - xyz has equal numbers of a's and b's, but y does not.
   - Therefore $x(y^2)z$ has unequal numbers of a's and b's.

   So $x(y^2)z \notin L$ because all strings in L have equal numbers of a's and b's.

   Example: If $xyz = aaaabbbb = (aaa)(abb)(bb)$, $x = aaa$, $y = abb$, $z = bb$, then y has unequal numbers of a and b. So $xyyz$ is $(aaa)(abb)(abb)(bb)$ and has 5 a and 6 b.

2. Suppose y has equal numbers of a and b.

   - Then, because $y \neq \epsilon$, y has at least one a and one b.
   - Now $x(y^2)z$ is $xyyz$, so y appears twice, so there is a b in the first y and an a in the second y.
   - So at least one b appears before one a in $x(y^2)z$.
   - However, all strings in L have all a before all b.

Therefore $x(y^2)z \notin L$.

Example: If $xyz = aaaabbbb = (aa)(aabb)(bb)$, $x = aa$, $y = aabb$, $z = bb$, then y has both a and b. So $xyyz$ is $(aa)(aabb)(aabb)(bb)$ and has a b before an a.

There is another way to do the proof. You can note that because $|xy| \leq N$, y consists only of a's, so that it is only necessary to consider the first case in the above proof.

1.2 The pumping theorem as a game

The pumping theorem can also be expressed as a game between two players. For this see handout 4.

- It doesn't matter who wins a particular game, but who has a winning strategy.
- If you have a winning strategy, the language is non-regular.
- If the opponent has a winning strategy, you don't know if the language is regular or not.

Note that the game can only show that a language is non-regular; it cannot show that a language is regular.

To show that a language is regular, you can find an automaton for the language.

For $L = \{a^n b^n\}$, here is the winning strategy for you.

1. The opponent chooses N.
2. You choose the string $a^N b^N$.
3. The opponent chooses x, y, and z such that $w = xyz$, $y \neq \epsilon$, and $|xy| \leq n$.
4. You choose $i = 2$.

As we argued above, this is a winning strategy for you. Note that it does not constrain what the opponent does; no matter what the opponent does, it gives you a response that leads to a win for you.

For $L = \mathcal{L}(a^*b^*)$, the opponent has a winning strategy. Note that this language can be recognized by a three state deterministic finite automaton. Here is the opponent's winning strategy:

1. The opponent chooses $N = 3$.
2. You choose a string of length 3 or more.
3. The opponent runs this string through his automaton for L and notes where it goes through the same state twice. The oppenent uses this to divide the string into x, y, and z such that $xy^iz \in L$ for all i.
4. You choose some i. Because $xy^iz \in L$, you lose.

Note that the opponent's winning strategy specifies his moves, but none of your moves.

For any regular language L, the opponent has a winning strategy.

- Because L is regular, there is a deterministic finite automaton M recognizing L.
- Let n be the number of states of M.
- Then the opponent's winning strategy for L is this:
    1. The opponent chooses $N = n$.
    2. You choose a string of length n or more.
    3. The opponent runs this string through his automaton for L and notes where it goes through the same state twice. The oppenent uses this to divide the string into x, y, and z such that $xy^iz \in L$ for all i.
    4. You choose some i. Because $xy^iz \in L$, you lose.

If the opponent is foolish, he can choose N smaller than n, or he may choose x, y, and z foolishly, and then you may be able to win the game. But

the important thing is who has a winning strategy if they play perfectly, not who wins a particular game.

There are some unusual non-regular languages for which the opponent may have a winning strategy. So if the opponent has a winning strategy, you don't know if L is regular or not.

Here are two non-regular languages for which the pumping lemma fails to show that they are non-regular:

1. $L_1 = \{a^i b^j c^k : i, j, k \geq 0, \text{ if } i = 1 \text{ then } j = k\}$.

   `http://www.cs.nthu.edu.tw/~wkhon/assignments/assign1ans.pdf`

2. $L_2 = \{c^m a^n b^n : m, n \geq 1\} \cup \{a, b\}^*$ (Kamala and Rama text)

How can the opponent win for language L_1, for example?

- If you choose a string containing at least one a then the opponent can choose a substring consisting entirely of a's, and you lose because no matter how you pump it, you get a string in L_1 and the opponent wins.
- If you choose a string containing only b and c then the opponent can choose a substring having only b in it or only c in it. No matter how you pump it, you get a string in L_1 and the opponent wins.
- Thus the opponent has a winning strategy even though L_1 is not regular.

Exercise: Show that L_1 is non-regular using the fact that if L is regular and x is a string, then $\{y : xy \in L\}$ is also regular, or else look at intersections with a regular language.

1.3 Finite languages

If the language is finite, the opponent chooses N larger than the length of the longest string in L. Because you can't choose a string of length N or more, the opponent wins by default.

1.4 Intersecting with a regular language

Sometimes it helps to intersect a language with a regular language to show non-regularity. For example, consider the language L consisting of all strings of a and b that contain the same number of a and b.

- Now, consider $L_1 = L \cap \mathcal{L}(a^*b^*)$.
- Then all strings in L_1 have the same number of a and b, because they are in L, and they also have all a before all b, because they are in $\mathcal{L}(a^*b^*)$.
- So what do these strings look like? They are strings of the form a^nb^n, and any string of the form a^nb^n is in L_1.
- Thus $L_1 = \{a^nb^n : n \geq 0\}$.

We showed already that $\{a^nb^n : n \geq 0\}$ is non-regular.

- Thus L_1 is non-regular.
- If L were regular, L_1 would be regular, because the intersection of two regular languages is regular.

Therefore L is not regular.

This gives a way to show languages are non-regular without having to use the pumping theorem or the game each time.

1.5 An interesting fact

If L is an arbitrary subset of $\mathcal{L}(a^*)$, then L^* is a regular language.

This is not easy to prove, by the way. Note that for $L \subseteq \Sigma^*$ where Σ has two or more letters, L^* need not be regular if L is not regular.

Example: Suppose $L = \{a^n : n \text{ is a perfect square}\}$. Thus $L = \{a, a^4, a^9, a^{16}, \ldots\} = \{a, aaaa, aaaaaaaaa, \ldots\}$. Then L is not regular. What is L^*? Is it regular?

Now suppose $L = \{a^n : n \text{ is a perfect square}, n > 1\}$. Thus $L = \{a^4, a^9, a^{16}, \ldots\} = \{aaaa, aaaaaaaaa, \ldots\}$. Then L is not regular. What is L^*? Is it regular?

Problem: Find a language L such that L^* is not regular.