A Note on Proving Non-Regularity via Fooling Sets

Chandra Chekuri*
University of Illinois at Urbana-Champaign

February 20, 2021

1 Introduction

This is a short note on the fooling set technique to prove that a given language L over a finite alphabet Σ is not regular. There are generally three techniques for this.

- Pumping lemma
- Fooling sets (or equivalently the Myhill-Nerode theorem)
- Closure properties

Most courses in CS curricula confine attention to the pumping lemma, and touch upon the use of closure properties (which is a reduction based approach). They generally do not teach the Myhill-Nerode theorem due to its abstractness, and the limited time available. Sipser's book on Theory of Computation relegates Myhill-Nerode theorem to a starred problem (he included this only in the second edition after feedback on his first edition). At University of Illinois, in the undergraduate course CS 374: Algorithms and Models of Computing, we teach the fooling set technique for proving non-regularity, even though we do not explicitly teach the Myhill-Nerode theorem. We refer the reader to notes on this topic by Jeff Erickson and Mahesh Viswanathan and also Sariel Har-Peled and Madhu Parthasarathy.

The purpose of this note is to provide a small addendum to the fooling set approach to make it easier for students to come up with a proof, without losing the advantages of the fooling set technique.

1.1 The Fooling Set Technique

For the sake of completeness we review the fooling set technique. We assume that the reader is already familiar with DFAs and notions of regularity and related concepts. A reader familiar with fooling sets can quickly refer to Lemma 1.5 and go directly to the next section.

Suppose L is a language over Σ, that is $L \subseteq \Sigma^*$. L is regular iff there is a DFA M such that $L = L(M)$. How do we prove that L is not regular? Every finite language is regular so necessarily L has to be infinite. DFAs do not have memory beyond their states. The main intuition behind proving non-regularity of a language L is that recognizing L requires memory that does not have a fixed finite bound. How do we formalize this? We play a game which is essentially a proof by contradiction. Suppose L is regular. Then there is a DFA M with some n states where n is finite. We then try to show that n states are not sufficient to recognize L. If we succeed in proving this we have shown that there is no DFA for L with at most n states. If the proof, like in induction, works for an *arbitrary* n then we have shown that there is no DFA for L.

*chekuri@illinois.edu. Comments and corrections are welcome.

How do we prove that a DFA with n states cannot recognize L. Let us try $n = 1$ as an exercise. A DFA with one state consists of its start state and self loops for every $a \in \Sigma$. It either accepts all strings if the start state is an accepting state, or accepts no strings if the start state is not an accepting state. Thus a 1 state DFA can only accept $\emptyset$ and Σ^*. Hence, if our given L is not one of these two simple languages we are done. Let us try $n = 2$. We can do brute force and try all machines with 2 states and classify the languages and check whether our L belongs to one of them. It is easy to see that this technique is not scalable. Instead we will work out a more abstract idea that will help us generalize.

Suppose we have two strings $x, y, x \neq y$. What will a DFA $M = (Q, \Sigma, \delta, s, A)$ do when we given them as input? The state that M will reach on x is $\delta^*_M(s, x)$, and the state that M will reach on y is $\delta^*_M(s, y)$. The nature of a DFA is such that if $\delta^*(s, x) = \delta^*(s, y)$ then the behaviour of M on xz and yz for any string z is the same. That is if $\delta^*(s, x) = \delta^*(s, y)$ then for all $z \in \Sigma^*$, $\delta^*(s, xz) = \delta^*(s, yz)$. This motivates the following definiton.

Definition 1.1. *Given a language L over a finite alphabet Σ, two strings $x, y \in \Sigma^*$ are suffix distinguishable with respect to L if there is a string $z \in \Sigma^*$ such that exacly one of xz, yz is in L. We say that z is a distinguishing suffix for x, y.*

Note that the definition is about the language L and the strings x, y. It is *not* about DFAs.

Example 1. *Let $L = \{0^i \mid i \mod 5 = 0\}$. Then 0 and 000 are distinguishable by suffix 00; $000 \notin L$ and $00000 \in L$. The strings 0 and 000000 are* **not** *distinguishable with repect to L.*

Exercise: Let $L = \{0^i \mid i \mod 5 = 0\}$. Prove that for $0^i, 0^j$ are distinguishable with respect to L if and only if $i - j \mod 5 \neq 0$.

Example 2. *Let $L = \{w \in \{0,1\}^* \mid w$ ends in $01\}$. Strings 00 and 1 are distinguishable by 1.*

Example 3. *Let $L = \{0^n 1^n \mid n \geq 0\}$. The strings 0^i and 0^j are distinguishable for any $i \neq j$.*

We will prove a seemingly silly lemma.

Lemma 1.2. *If L has a distinguishable strings x, y then no DFA M with one state can recognize L.*

Proof: Suppose M has only one state. Then $\delta^*(s, x) = \delta^*(s, y)$ since they both have to be equal to s. But this means that M can no longer tell apart x and y. But since they are a distinguisable pair there is a z such that exactly one of xz and yz is in L but we have $\delta^*(s, xz) = \delta^*(s, yz)$ which implies that M will accept both or reject both but this means M does not correctly recognize L. □

In fact a more general statement is the following whose proof is very similar to the one above.

Lemma 1.3. *If L has a distinguishable strings x, y and $M = (Q, \Sigma, \delta, s, A)$ is any DFA that recognizes L then $\delta^*(s, x) \neq \delta^*(s, y)$.*

How do we leverage the preceding observation? The key definition is that of a *fooling set*.

Definition 1.4. *Let L be a language. A set of strings F is a fooling set for L if* **every** *pair of distinct strings in F is distinguishable with respect to L.*

The definition of a fooling set may seem innocuous but the condition is strong. *Every* pair of strings in F must be distinguishable. Note that each pair in F can be distinguished by a different suffix.

Exercise: Let $L = \{0^i \mid i \mod 5 = 0\}$. Prove that $\{\varepsilon, 0, 00, 000, 0000\}$ is a fooling set for L.

Exercise: Let $L = \{0^i \mid i \mod 5 = 0\}$. Prove that F is a fooling set for L if and only if the following is true: for any $x, y \in F$, $|x| - |y| \neq 0 \mod 5$.

Exercise: Let $L = \{w \in \{0,1\}^* \mid w \text{ ends in } 01\}$. Prove that $\{\varepsilon, 0, 01\}$ is a fooling set for L.

The following is not hard to prove from Example 3.

Exercise: Let $L = \{0^n 1^n \mid n \geq 0\}$. Prove that the infinite set $F = 0^*$ is a fooling set for L.

The key lemma is the following.

Lemma 1.5. *Let L be a language and let F be a fooling set for L. No DFA M can recognize L if it has less than $|F|$ states. In particular, if $|F|$ is infinite then L cannot be regular.*

Proof: The proof is a simple extension of the basic lemma we saw earlier and is by contradiction. First we prove the case when $|F|$ is finite. Let $F = \{w_1, w_1, \ldots, w_n\}$. Let $M = (Q, \Sigma, \delta, s, A)$ be any DFA with less than n states, that is $|Q| < n$. For $i = 1$ to k let $q_i = \delta^*(s, w_i)$ be the state that M reaches on input w_i. Since $|Q| < n$ there must be two distinct strings $w_i, w_j \in F$ with $i < j$ and $q_i = q_j$; this follows from the pigeon hole principle. In other words the machine M will be in the same state after seeing w_i and w_j. However w_i and w_j are distinguishable with respect to L which means that there is a suffix z such that exactly one of $w_i z, w_j z$ is in L; however, M will take both $w_i z$ and $w_j z$ to the same state and hence will accept both or reject both, thus making a mistake. Hence F fooled M and M could not be a correct DFA for L.

Now consider the case that F is infinite. Suppose L is regular. This implies that there is some n such that L has a DFA with n states where n is finite. Since F is infinite we can pick a subset $F' \subset F$ such that $|F'| = n + 1$. Note that any subset of a fooling set for L is also a fooling set for L, hence F' is a fooling set of size $n + 1$ for L. By the argument in the preceding paragraph M will make a mistake on L since $n < |F'|$. This contradicts the assumption that L had a DFA with n states. □

From the preceding lemma we see that if L has a fooling set of size n then it establishes a lower bound of n on the number of states in any DFA that can recognize L. In particular if F is infinite then L is not regular.

Exercise: Let $L = \{w \in \{0,1\}^* \mid w \text{ ends in } 01\}$. Since $\{\varepsilon, 0, 01\}$ is a fooling set for L any DFA that recognizes L must have at least 3 states. Prove that there is a DFA with 3 states.

Claim 1. *The language $L = \{0^n 1^n \mid n \geq 0\}$ is not regular since 0^* is an infinite fooling set for L.*

A very natural question one can ask is the following. If L is not regular can we always find an infinite fooling set? In other words, does this method always work to prove non-regularity. Indeed this is true and captured by the Myhill-Nerode theorem.

Theorem 1.6 (Myhill-Nerode). *Let L be any language. Then*

- *If L is not regular then there is an infinite fooling set for L.*
- *If L is regular then there is a fooling set F of size k where k is the smallest number of states of a DFA that accepts L.*

1.2 Coming up with Fooling Sets

How does one come with fooling sets for a given language L? We are here primarily concerned with the task of proving that a language is not regular by coming up with an infinite fooling set. Coming up with an infinite fooling set is often non-obvious. We make a simple but useful observation on fooling sets that is often helpful in getting started.

Lemma 1.7. *Suppose F is a fooling set for L. Every string in F except for one must be a prefix of some string in L. In particular, if L is non-regular there is an infinite fooling set consisting of only prefixes of strings in L.*

In other words, when thinking of infinite fooling sets for a language L, it suffices to confine attention to prefixes of strings in L.

Proof: Suppose $x, y \in F, x \neq y$ where both x and y are both not prefixes of strings in L. We claim that x, y are not distinguishable. Distinguishable means that there is a string z such that exactly one of xz, yz is in L. But if xz is in L then x is a prefix of a string in L and similar if $yz \in L$, y is a prefix. □

The purpose of this note is to provide a slightly simpler way to find fooling sets via the following lemma.

Lemma 1.8. *A language L is non-regular if and only if for every $n \geq 1$ there a is a fooling set F_n for L where $|F_n| > n$.*

Proof: We sketch the proof since it easily follows from what we have seen so far. If L is not regular then by the Myhill-Nerode theorem there is an infinite fooling set F. This easily implies that we can construct a fooling set F_n of size $n+1$ by taking a subset of F.

For the converse we refer to Lemma 1.5. □

What advantage does this formulation give? The reason why this seemingly simpler formulation has some advantage is that it lets one explore more systematically how to find a fooling set of size $> n$ by considering some fixed n; the finitary aspect of this is useful in our opinion. Moreover it reminds the prover explicitly about the reasoning behind the proof: a language is non-regular iff we can find larger and larger fooling sets.

We illustrate with some examples.

Consider the canonical non-regular language $L = \{0^k 1^k \mid k \geq 0\}$. Fix some n. We claim that $F_n = \{\varepsilon = 0^0, 0, 00, \ldots, 0^n\}$ is a fooling set of size $n+1$ for L. Note that we just need to come up with a finite fooling set since n is finite. We can easily see in this case that $F_n \subset F_{n'}$ if $n < n'$ and that $F = 0^*$ is an infinite fooling set for L. The point is that it is *not* necessary to actually figure out F to prove that L is non-regular. Once we create F_n for each n such that $|F_n| > n$ we are done.

Now we consider a more involved example. Consider $L = \{0^{k^2} \mid k \geq 0\}$. Why is this non-regular? Coming up with an infinite fooling set for L may be difficult right away. However, let us start the process by fixing n and asking how we can find a fooling set of size $> n$. A natural starting point is to start with the set $F_n = \{\varepsilon, 0, 00, \ldots, 0^n\}$ which has size $n+1$ since the strings are all prefixes of strings in L. But then we need to prove that *any* pair of strings in F_n are distinguishable with respect to L. How do we prove this? We need to show that if $0 \leq i < j \leq n$ then there is some string 0^h such that exactly one of $i+h, j+h$ is a square of an integer. This seems non-obvious. It is not hard to see that if we take small values of i, j we can do this by trial and error but question is how do we can prove it for all pairs. How do we find an easier fooling set of sufficiently large size? A heuristic approach is the following. If we believe

that L is non-regular it is because it is infinite and longer strings would require more memory. Here, having the concrete parameter n is useful. We should try looking for strings whose length is somehow a function of n and we also want more than n strings. Given the language L it is worth looking at strings longer than 0^{n^2}. We note that $0^{n^2} \in L$ and the next string is $0^{(n+1)^2}$. Thus we have a large gap between the length of two consecutive strings in L, more precisely, it is $(n+1)^2 - n = 2n+1$. The DFA M with $\leq n$ states should not accept any string with length strictly between n^2 and $(n+1)^2$ but has to accept 0^{n^2} and $0^{(n+1)^2}$. It is plausible to believe that it may have some difficulty counting to $2n+1$ with only n states. Based on this intuition we can try the following as a fooling set: $F_n = \{0^{n^2}, 0^{n^2+1}, \ldots, 0^{(n+1)^2-1}\}$. You should verify that indeed F_n is a fooling set for L and moreover $|F_n| \geq 2n$.

What is an infinite fooling set for L? It turns out that L is itself a fooling set for L. One can indeed verify this. But verifying is not the same as coming up with it. Note that the fooling set we came up for each n is quite different from L itself. This illustrates that the two different approaches can lead to different ways of thinking which is always a good thing in mathematics.

Exercise: Prove that $L = \{0^{2^k} \mid k \geq 0\}$ is not regular.

One can show that L is a fooling set of L in the preceding exercise but it is not so obvious or intuitive. However, using the same reasoning as the one for the language $\{0^{k^2} \mid k \geq 0\}$ find a fooling set F_n for L with $|F_n| > n$ for each $n > 0$.

Exercise: Prove that $L = \{0^{\lfloor k\sqrt{k} \rfloor} \mid k \geq 0\}$ is not regular.

Exercise: Prove that $L = \{0^{\lfloor k \log k \rfloor} \mid k \geq 1\}$ is not regular.

In fact we can prove a general lemma which simultaneously shows that many counting languages are not regular.

Lemma 1.9. *Let $L = \{0^{f(k)} \mid k \geq 0\}$ for some non-negative integer valued increasing function $f : \mathbb{Z}_+ \to \mathbb{Z}_+$ such that for any $n > 0$ there is some h such that $f(h+1) - f(h) > n$. Then L is not regular.*

Proof: For any n let h be an integer such that $f(h+1) - f(h) > n$. Consider the set $F_n = \{0^{f(h)}, 0^{f(h)+1}, \ldots, 0^{f(h+1)-1}\}$. It is easy to verify that F_n is a fooling set since f is increasing. □

An interesting corollary is the following. Let PRIMES $= \{0^p \mid p \text{ is a prime}\}$. We claim that PRIMES $= \{0^p \mid p \text{ is a prime}\}$ is not regular. What the preceding proof technique suggests is the following. Given any integer n can we find two consecutive primes p_1 and p_2 such that $p_2 - p_1 > n$? Indeed this is true via the following construction. Consider $n! + 1$. Let p_1 be the largest prime that is less than equal to $n! + 1$ and p_2 be the smallest prime that is $> n! + 1$. Note that by our definition, p_1 and p_2 are consecutive primes. We claim that each of the numbers $n! + 2, n! + 3, \ldots, n! + n$ is a composite number since $n! + i$ is divisible by i. Thus, $p_1 \leq n! + 1$ and $p_2 > n! + n$ and this implies that $p_2 - p_1 \geq n$.

Distilling the thought process: Here we attempt to distill the process of finding a fooling set of increasing size. Given language L and an integer n we are seeking to find a fooling set F_n for L such that $|F_n| \geq n$. First we try to find one distinguishing pair $x, y \in L$ such that both x, y are long strings. By long we are trying to find strings whose length depends on n somehow. Why? Because we need to eventually find n strings in F_n, the strings have to be necessarily long, so we may as well start with trying to find two long strings first. This process also gives us some feel for why the language may not be regular and what kind of information a machine may need to keep track of to correctly recognize L. In fact it may be helpful to think of a regular C program that can recognize L. Once we find two strings that are distinguishable we explore the possibility of finding more strings that are related to those two strings in some simple way. Here we keep in mind Lemma 1.7 which shows that we should find prefixes of strings in the language.

1.3 Pumping Lemma

Here we briefly describe the pumping lemma technique, mainly to contrast it with the fooling set technique and to see why it is sometimes simpler.

The high level approach is to prove that given L there is no DFA for L with n states for any n. Instead of trying to find a general fooling set F with $|F| > n$ the technique is based on cleverly finding a sufficiently long string r where $|r| \geq n$. Why? If $|r| \geq n$, since M has only n states it will have a "cycle" while processing r. More formally, let $r = a_1 a_2 \ldots a_h$ where $h \geq n$. Let $q_i = \delta^*(s, a_1 a_2 \ldots a_i)$ be the state that M reaches after seeing the prefix of r with the first i characters (including $i = 0$ which corresponds to ε). Then $q_i = q_j$ for some $0 \leq i < j \leq n$ since M has only n states and we have $n+1$ prefixes (pigeon hole principle). In other words the DFA goes from q_i to q_i (a cycle) when given the string v. It is then easy to observe that DFA M will go from q_i to q_i on any number of copies of v (it will cycle around). That is $\delta^*(q_i, v^\ell) = q_i$ for any $\ell \geq 0$. This also implies that M behaves the same way on all strings of the form $uv^\ell y$ for $\ell \geq 0$. Note that $|v| \geq 1$, otherwise it is not interesting at all. The name *pumping* comes from the fact that v can be pumped as many times as we want. We can write $r = uvw$ where $u = a_1 a_2 \ldots a_{i-1}$ and $v = a_i a_{i+1} \ldots a_j$ and w is rest of the string (w can be ε). Note that $|v| \geq 1$ which means that it is non-trivial and also $|uv| \leq n$. We can capture the preceding observation in the following lemma.

Lemma 1.10. *Suppose L has a DFA M with n states. Let r be any string such that $r \geq |n|$. Then r can be written as uvw for some u, v, w such that (i) $|v| \geq 1$ (ii) $|uv| \leq n$ and (iii) for any $\ell \geq 0$ the strings r and $uv^\ell w$ are* **not** *distinguishable.*

An important note is that the split of r into uvw is not under "our" control.

How can we exploit the above lemma to prove non-regularity of a given language L? We pick a clever string r. Suppose we can prove that for some ℓ, that we get to choose, the strings $x = uvw$ and $y = uv^\ell w$ have the property that exactly one of them is in L. But the lemma says that they are not distinguishable! Then we obtained the desired contradiction that L has a DFA with at most n states. Proving that exactly one of the string $x = r$ and $y = uv^\ell w$ is in L is the same as saying that they are distinguishable via the suffix ε. In fact it is sufficient to ensure that x, y are distinguishable by some suffix but that would be an added level of complexity so the basic technique insists on the simpler condition.

Let us illustrate this with a couple of examples before we formalize the technique.

Consider $L = \{0^k 1^k \mid k \geq 0\}$. Given n we have to first choose a string r such that $|r| > n$. Let us try $r = 0^{n+1}1^{n+1}$. Then if some M with $\leq n$ states accepts L we can write $r = uvw$ with $|v| \geq 1$ and $|uv| \leq n$. Since $|uv| \leq n$ and $r = 0^{n+1}1^{n+1}$ we must have $|uv| = 0^p$ for some $p \leq n$. Now consider $uv^\ell w$. What is it? If we set $\ell = 0$ we have $y = uv^0 w$. We note that $r \in L$ while $y \notin L$ since y is obtained from r by removing some $|v|$ zeroes from r which means that in the resulting string the numbers of 0's and 1's are different. Thus, $y \notin L$. In fact it is easy to see that $x = r \in L$ and $y = uv^\ell w \notin L$ for any $\ell \neq 0$. Since the proof works for any n, this proves that L is not regular. Remember that we did not have any choice on what p is because the pumping lemma only guarantees that r can be written as uvw for some uv with $|v| \geq 1$ and $|uv| \leq n$. However our proof showed that any such v will work to get a contradiction.

A second example. Consider $L = \{0^{k^2} \mid k \geq 0\}$. What should we choose r for a given n? We will choose $r = 0^{n^2}$. In general we want to choose a string that is long compared to n and this is a natural and reasonable choice. We have $r = uvw$ for some $|v| \geq 1$ and $|uv| \leq n$. Let us look at $y = uv^2 w$. We have $y = 0^{n^2}0^p$ where $p = |v|$. We have $1 \leq p \leq n$. We observe that $n^2 + p > n^2$ and $n^2 + p < (n+1)^2$ since $(n+1)^2 - n^2 = 2n + 1 > p$. This means that $y \notin L$. Thus $x = r \in L$ and $y \notin L$ giving us the desired contradiction.

We now formalize the pumping lemma technique for proving non-regularity of a given language L.

- Choose cleverly a string r_n for an arbitrary integer n such that $|r_n| \geq n$.

- Assume that r_n can be written as uvw where $|v| \geq 1$ and $|uv| \leq n$. Note that the split is not under our control so we can only use the properties about length of v and uv.
- Show that one can choose some $\ell \neq 1$ such that r_n and the string $uv^\ell w$ are distinguishable with respect to L thus obtaining a contradiction to the pumping lemma. Typically we choose $r_n \in L$ and find some $\ell \neq 1$ such that $uv^\ell w \notin L$.

The cleverness in using the pumping lemma comes from picking the string r_n for a given n and then choosing ℓ appropriately in the third step to prove the contradiction. It is problem specific.

Pumping lemma technique has the advantage of focusing on finding a *single* clever string from the language L. When it works, which happens for many interesting examples, it is a fairly simple technique. The disadvantage of the pumping lemma is that it doesn't always work to prove non-regularity while the fooling set technique is fool-proof (sic) via the Myhill-Nerode theorem. Anoter disadvantage of the pumping lemma technique is that students often find the lemma and the quantifiers confusing and misunderstand it. However, there is no universal agreement on whether one should teach the pumping lemma over the fooling set technique. Ideally we should teach both but time is short and most courses stick to one or the other.

Exercise: Prove that the languages $\{0^{2^k} \mid k \geq 0\}$ is not regular.

Exercise: Prove Lemma 1.9 using the pumping lemma.