CSE396 Lecture Tue. Feb. 17: Regular and Non-Regular Languages, Myhill-Nerode Theorem

John Myhill, UB Math Department: 1966 until his death in 1987.
Anil Nerode, Cornell Mathematics and MSI. My postdoc overseer 1986--89. Still active!
Theorem proved while both were at U.Chicago in 1958.

The intuition is that a language L is non-regular exactly when there are infinitely many status possibilities you have to "juggle in your mind" as you process a string x left-to-right. [The "Pumping Lemma" has a different idea and doesn't apply "exactly"---another reason I prefer MNT.]

For example, suppose we liberalized the rules of the "spears and dragons" game to the ridiculous extent that you could carry as many spears as you find. Then you would have a status of "n spears" for every natural number n. Strings of the form $\$^n D^n$ are now "survivable" for every n. But the language L_∞ that allows these strings is not regular.

Well, we have to prove that. The idea of the proof is that the reason different numbers of spears, say $x = \$^m$ and $y = \$^n$ where $m < n$, are different statuses is that if n rooms with dragons come next (symbolized by the string $z = D^n$), then the x status gets you killed whereas the y status survives. That is, $xz \notin L_\infty$ but $yz \in L_\infty$. Put another way, $L_\infty(xz) \neq L_\infty(yz)$, where we've taken the liberty of regarding a language as a Boolean function that returns `true` on strings that belong to it and `false` on strings that don't.

Let's build up this idea from the point of view of a language A that ***is*** regular, using a DFA M such that $L(M) = A$ and seeing what states it needs in order to process various strings correctly.

Myhill-Nerode Technique -- Quick Version

----

1. Suppose you are trying to build a DFA M to recognize a language A and you find three strings x, y, z such that $xz \in A$ but $yz \notin A$ or vice-versa. Then your M must process x and y to different states. The reason is that if you had M process them to the same state q, then upon following up with processing z from state q, both xz and yz would end up in the same state, call it r. But r can't be both an accepting and a rejecting state.

2. Let us abbreviate "$xz \in A$ but $yz \notin A$ or vice-versa" as $A(xz) \neq A(yz)$. Then write $\boldsymbol{x} \nsim_A \boldsymbol{y}$ if there exists a z such that $A(xz) \neq A(yz)$. Call x and y **pairwise distinguishable (PD) for A** when that happens. Merely re-phrasing what we showed in step 1, if x and y are PD for A then any DFA M that attempts to make $L(M) = A$ must process x and y to different states.

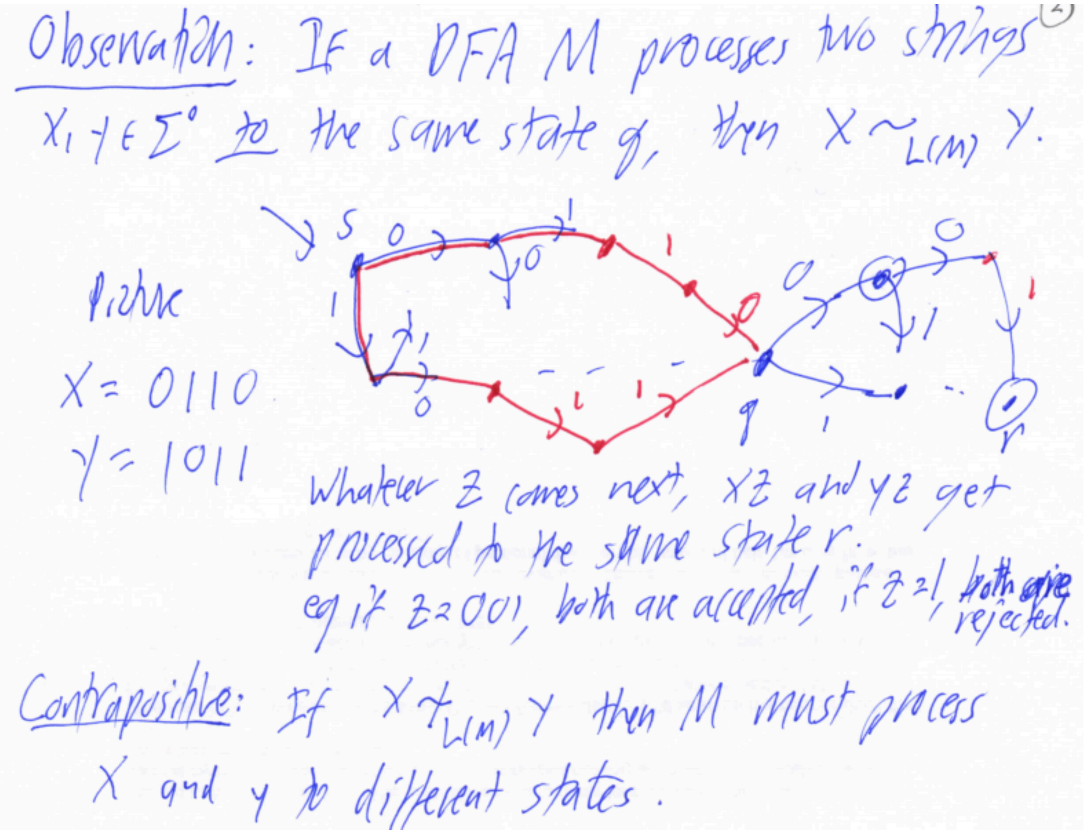
3. Therefore, if we have a set S of k strings, any two of which are PD for A, then any such M needs to process them to k different states. We call S a **PD set** for A.

4. Therefore, if we have a set S of ∞-many strings, any two of which are PD for A, then any such M needs to process them to ∞ different states.

5. But having infinitely many states contradicts the definition of M being a deterministic **finite** automaton. Hence, a DFA M such that $L(M) = A$ cannot exist. So A is not a regular language.

This yields a technique for proving a language A to be non-regular (when it's true): find an appropriate infinite set S and prove that S is PD for A. The "proof script" for applying the technique will be even shorter than this.

But first, let us go over the steps more carefully. For **step 1**, here is a picture:



Observation: If a DFA M processes two strings $x, y \in \Sigma^*$ to the same state q, then $x \sim_{L(M)} y$.

Picture

$x = 0110$

$y = 1011$

Whatever z comes next, xz and yz get processed to the same state r.

eg if $z = 001$, both are accepted, if $z = 1$, both are rejected.

Contrapositive: If $x \not\sim_{L(M)} y$ then M must process x and y to different states.

OK, we wrote $x \sim_{L(M)} y$, and saying what this means takes us into **step 2**. For a general language A, we define:

> **Definition**: Two strings x, y are A-equivalent, written $\boldsymbol{x \equiv_A y}$, if for all $z \in \Sigma^*$,
> $$xz \in A \iff yz \in A.$$

We can abbreviate this to: $x \sim_A y$ if $(\forall z) A(xz) = A(yz)$. The negated relation is just:

$$x \not\equiv_A y \text{ if } (\exists z) A(xz) \neq A(yz).$$

An example of equivalent strings for a regular language is $E = \left\{x \in \{0,1\}^* : \#1(x) \mathit{\ is\ even}\right\}$. Then $x = 100$ and $y = 10101$ are equivalent, because they have the same parity (odd) of number of 1's. But $x = 100$ and $y' = 1001$ are not equivalent: $x \nsim_E y'$. This is why the DFA we saw for E must go from its "odd 1s" state back to its "even 1s" start state on the final 1 in y'. A simple example of equivalent strings for a nonregular language is that if we are interested in prime numbers that begin with the digit 7, the prefixes $x =$ "7" and $y =$ "007" are equivalent, because leading zeroes don't change the number whatever comes after the 7.

Although we don't need to employ this since "PD" uses only the negated relation $x \nsim_A y$, it helps to observe that for any language A, regular or not, $\sim_A$ is an **equivalence relation**.

1. **Reflexive**: $x \sim_A x$ since obviously $(\forall z)A(xz) = A(xz)$.
2. **Symmetric**: $x \sim_A y \iff y \sim_A x$ since $(\forall z)A(xz) = A(yz)$ is the same thing as $(\forall z)A(yz) = A(xz)$.
3. **Transitive**: we need to show $(w \sim_A x \wedge x \sim_A y) \implies w \sim_A y$. The left-hand sides say:
   - $(\forall z)A(wz) = A(xz)$
   - $(\forall z')A(xz') = A(yz')$.

   Since the two quantifiers $(\forall z)$ and $(\forall z')$ run over the same strings, the equations add up to
   - $(\forall z)A(wz) = A(yz)$,

   which yields $w \sim_A y$.

The extra thing this lets us say is that when $x \nsim_A y$, this means x and y belong to different equivalence classes. Maybe this helps us re-state **step 3**:

3'. If we have a set S of k strings, any two of which are PD for A, then they fall into k different equivalence classes. Since step 1 says a DFA M giving $L(M) = A$ would need different states for different equivalence classes, it follows that M needs (at least) k states, one for each equivalence class.

If going from "any 2 are different" to "all k are different" is clear to you, great. My own MNT handout with horses tries to make this step obvious by saying: If two horses are distinctive, color them different colors. Hence if every pair of k horses are distinctive, they all have k different colors. Thus a DFA, needing a different state to process each color to, must have at least k states.

Well, the historical way to reason about it involves pigeons rather than horses:

> Suppose M had $k-1$ (or fewer) states. Then among the k strings $x_1, \dots, x_k$ in the set S, (at least) two of them must get processed to the same state. They are like k "pigeons" trying to be placed into $k-1$ "holes" in a pigeon carrying case---you'd have to try to put two of them in the same slot. Call those two x_i and x_j. By the definition of S being PD for A, we have $x_i \nsim_A x_j$. But then by step 1, x_i and x_j have to go to different states, a contradiction. So and DFA M such that $L(M) = A$ must have at least k states.

The key point is called the **Pigeonhole Principle**. Originally it was stated in the 1830s by the French mathematician Johann Peter Gustav Lejeune Dirichlet in terms of k pearls going into $k-1$ drawers of a jeweler's carrying case. IMHO the "pigeon" analogy took hold not only in France because pigeon carriers are similar cases with square slots, but even more in England because the square pass-through mail slots in the Porter's Lodge of a residential college are called *pigeonholes*. The college mail service itself is called the "Pigeon Post." The principle says that if k letters arrive for $k-1$ residents of the college, then some lucky fellow will get more than one letter.

[See recent short GLL blog post https://rjlipton.wordpress.com/2021/02/15/pigenhole-principle/]

[In the "Pumping Lemma", the same principle is applied to say while processing the first k characters of a string x in the language, a machine with $k-1$ states must have been in the same state q at least twice. The substring w that got processed between the two times it was in state q can then be either erased ("pumping it down") or repeated any number of times ("pump up") so as to leave another string x' that also must be in the language...but...contradiction...so no machine.]

Step 4: If we have a set S of infinitely many PD strings for A, then any DFA M such that $L(M) = A$ would need infinitely many states---but that is a contradiction in terms. For if M had a finite number k of states, then taking just $k+1$ strings from S would already put us in "pigeonhole trouble."

Step 5: Therefore we have proved the following:

Myhill-Nerode Theorem (MNT), first part: If **there is** an infinite PD set for a language A, then A is not regular. ☒

[The second part is the *converse*: if a language A is not regular, then there is an infinite PD set. Thus, unlike the Pumping Lemma, the MNT gives an exact characterization of (non-)regular languages. Whereas some languages cannot be proved nonregular directly by the Pumping Lemma, in principle there is always a proof by MNT. The proof of the second part actually shows the *inverse* (which is the *contrapositive* of the *converse*): if all PD sets for A are finite, then A *is* regular. Put another way, A is regular if and only if the relation $\sim_A$ has only finitely many equivalence classes. The idea of proving the inverse direction is that the equivalence classes actually become states of a DFA $M_A = (Q_A, \Sigma, \delta_A, s_A, F_A)$. The start state s_A is the equivalence class of ϵ and the final states F_A are the equivalence classes of those strings that belong to A. There are ∞-many strings in Σ^* but (as with the #1(x)-even and #1(x)-odd example), that doesn't stop the possibility of there being only finitely many equivalence classes. The δ_A part is a little trickier to define, but...this is not the part that gets applied, anyway.]

Applying MNT

There are two steps, of which the first is quick once you get the idea.

1. Choose (wisely) an infinite set S. (It need not be a subset of the language A we are trying to prove non-regular. It is often intuitively a set of "critical initial segments" of strings in A.)
2. Prove that S is PD for A. If we unroll the logic of the definition of PD, we want to prove:

for all $x, y \in S\ (x \neq y)$, there exists $z \in \Sigma^*$ such that $A(xz) \neq A(yz)$.

We can lay out what we need to do in any given case in a template for a "proof script":

Take $S =$ ______________________. "Clearly S is infinite."

Let any $x, y \in S$ (such that $x \neq y$) **be given**. Then we can helpfully write $x =$ __________ and $y =$ __________ where _______________ [optional: where _____________ "wlog."]

Take $z =$ _______________. Then $A(xz) \neq A(yz)$ because __________________ ____.

Since x and y are an arbitrary pair from S, S is PD for A, and since S is infinite, A is nonregular by the Myhill-Nerode Theorem. ☒.

Example 1: $A = \{0^n 1^n : n \geq 0\}$.

Take $S =$ _________ 0^* ______________. "Clearly S is infinite."

Let any $x, y \in S$ (such that $x \neq y$) **be given**. Then we can helpfully write $x =$ ___ 0^m ____ and $y =$ ____ 0^n ______ where ____ $m \neq n$ _____. [optional: where _________ "wlog."]

Take $z =$ ____ 1^m ______. Then $A(xz) \neq A(yz)$ because _____ $xz = 0^m 1^m$ which is in A, but $yz = 0^n 1^m$ which is not in A since $m \neq n$ ______.

Since x and y are an arbitrary pair from S, S is PD for A, and since S is infinite, A is nonregular by the Myhill-Nerode Theorem. ☒.

Example 2: $A = \{x \in \{0,1\}^* : x = x^R\}$. Here x^R means x reversed. For example, $011^R = 110$. And $\epsilon^R = \epsilon$. If $x = x^R$ then x is a **palindrome**. E.g.: $010, 1001, 0000000$. Let's focus on the palindromes $1,\ 010,\ 00100,\ 0001000,\ 000010000,\ \ldots$

$L_a = \{xx^R\}$

Take $S = $ __________ $(00)^*1$ _______________. "Clearly S is infinite."

Let any $x, y \in S$ (such that $x \neq y$) **be given**. Then we can helpfully write $x = $ ___ $(00)^m 1$ ____ and $y = $ ____ $(00)^n 1$ _______ where _____ $m \neq n$ _____. [optional: where __________ "wlog."]

Take $z = $ ____ $1(00)^m$ ______. Then $L_a(xz) \neq L_a(yz)$ because ______ $xz = (00)^m 11(00)^m$ which is in L_a, but $yz = (00)^n \cdot 11(00)^m$ which is not in L_a since $m \neq n$ ______.

Since x and y are an arbitrary pair from S, S is PD for L_a, and since S is infinite, L_a is nonregular by the Myhill-Nerode Theorem. ☒.

Notice that the only edits to the previous proof were the three insertions of 0 marked in pink. The standard name for this language is PAL.

The next example requires using the "without loss of generality" ("wlog.") feature.

Example 3: $A = \left\{x \in \{0,1\}^* : \#0(x) \leq \#1(x)\right\}$. (Note, incidentally, that if it said ' = ' then we would be able to re-use the first proof verbatim. Here it is before we change it...)

Take $S = $ __________ 0^* _______________. "Clearly S is infinite."

Let any $x, y \in S$ (such that $x \neq y$) **be given**. Then we can helpfully write $x = $ ___ 0^m ____ and $y = $ ____ 0^n _______ where _____ $m \neq n$ _____. [optional: where __________ "wlog."]

Take $z = $ ____ 1^m ______. Then $A(xz) \neq A(yz)$ because _____ $xz = 0^m 1^m$ which is in A, but $yz = 0^n 1^m$ which is not in A since $m \neq n$ ______.

Since x and y are an arbitrary pair from S, S is PD for A, and since S is infinite, A is nonregular by the Myhill-Nerode Theorem. ☒.

But for ' $\leq$ ' we have to fix the proof, because if $m > n$ then $yz = 0^n 1^m$ would still be in A. The point is that given two strings of 0's, which have to be different lengths if they are in S, we can freely suppose the "m" refers to the shorter one and "n" to the longer one. There is no loss of generality in doing so. Thus, we can assert "wlog. $m < n$." That is enough to make the entire rest of the proof work without changing a thing, since the assertion guarantees that $yz = 0^n 1^m$ is not in A.

$A = \left\{x \in \{0,1\}^* : \#0(x) \leq \#1(x)\right\}$

Take $S = $ __________ 0^* _______________. "Clearly S is infinite."

Let any $x, y \in S$ (such that $x \neq y$) **be given**. Then we can helpfully write $x = $ ___ 0^m ____ and $y = $ ____ 0^n _______ where _____ $m < n$ wlog.

Take $z = $ ____ 1^m ______. Then $A(xz) \neq A(yz)$ because _____ $xz = 0^m 1^m$ which is in A, but $yz = 0^n 1^m$ which is not in A since $m < n$ ______.

Since x and y are an arbitrary pair from S, S is PD for A, and since S is infinite, A is nonregular by the Myhill-Nerode Theorem. ☒.

[do as many examples as time allows, picking up with more on Thursday]

Example 4: Suppose we "upgrade" the spears-and-dragons game to allow the Player to hold arbitrarily many spears. The Player still loses a spear for each dragon killed. Recall the alphabet is $\{0, \$, D\}$ with 0 standing for empty room, $\$$ for spear, and D for dragon. We can do the proof that the resulting language A is nonregular even without specifying exactly what A is, and while ignoring the presence of 0.

Take $S =$ __________ $\$^*$ _______________. "Clearly S is infinite."

Let any $x, y \in S$ (such that $x \neq y$) **be given**. Then we can helpfully write $x =$ ___ $\$^m$ ____ and $y =$ _____ $\$^n$ _______ where _____ $m < n$ wlog.

Take $z =$ _____ D^n _______. (Note change from m to n here.) Then $A(xz) \neq A(yz)$ because ______ $xz = \$^m D^n$ which is not in A because $m < n$ so the Player gets killed, but $yz = \$^n D^n$ which is in A since the Player kills exactly the possible number of dragons and survives with zero spears left over _______.

Since x and y are an arbitrary pair from S, S is PD for A, and since S is infinite, A is nonregular by the Myhill-Nerode Theorem. ☒.