Def: Let $x, y$ be strings and $L$ be a language. We say that $x$ and $y$ are **indistinguishable by $L$** if for every $z$ the following holds: $xz \in L$ iff $yz \in L$. We write $x \equiv_L y$.

Note: this is an **equivalence** relation.

Examples: find the equivalence classes of $\equiv_L$:

1. $L_1 = \{ 0w \mid w \in \{0,1\}^* \}$
   - Equiv. classes: $[0] =$ all strings starting $w$, $0 \in$ class 1,
   - $[1] =$ all strings starting $w$, $1 \in$ class 2,
   - $[\varepsilon] =$ remainder $2$.

2. $L_2 = \{ w \in \{0,1\}^* \mid \text{sum of digits of } w \text{ is divisible by 3} \}$
   - Equiv. classes: $[\varepsilon] =$ all strings $w$, sum of digits of $w$ divisible by $3$,
   - $[1] =$ all strings $w$, sum of digits of $w$ and $3d = 1$,
   - $[0] =$ remainder $2$.

3. $L_3 = \{ 0^k1^k \mid k > 0 \}$
   - Equiv. classes: $[0^i] \subset$ contains only $0^i$. 

**Note:**
- $3 \not\equiv_L 01$ (we want to say no).
- $110 \not\equiv_L 01$ (we need to find $z$ s.t. $110z \in L$ and $01z \not\in L$, or vice versa).
- Let's take $z = \varepsilon$.
- $\varepsilon \equiv_L x$ s.t. the sum of digits in $x$ is divisible by $3$.
- Let's consider $z$:
  - $x \equiv_L z \iff$ sum of digits of $z$ is divisible by $3$.
  - $xz \equiv_L yz \iff$ sum of digits of $z$ is divisible by $3$. 

**Proof:**
- $z \equiv_L \varepsilon$.
Myhill-Nerode Thm

$L_3 = \{ \, 0^k1^k \mid k > 0 \, \}$

We want to show that $0^j$, for $j \geq 0$, are all lonely elements of their equiv. classes

$[0^j]$ - contains only $0^j$

Suppose, by contradiction, there is $x \neq 0^j$ in $[0^j]$

Then $x = 0^j0^j$ but now consider $z = 01^j+1$

We get $0^jz = 0^j1^j1^j+1 \in L_3$

What about $xz = \overline{011 \ldots 1_j+1}$

\[\text{infinite \# equiv. classes}
\text{(there are other equiv. classes than } [0^j])\]

We'll see that a language $L$ is regular iff #equiv. classes is finite

\[\text{Myhill-Nerode}\]

\[\text{to be in } L_3, \text{ the block of } j+1 \text{ ones must be}
\text{preceded by } j+1 \text{ zeros}
\text{Thus, the only } x \text{ for which } xz \in L_3 \text{ is } x = 0^j
\text{but we are looking for } x \neq 0^j
\Rightarrow \text{there is no such } x\]
Consider a DFA accepting $L$. Suppose that $x$ and $y$ end in the same state $q$. What can we say about $x,y$?

Claim: If $L$ is accepted by a DFA with $\leq k$ states, then $\equiv_L$ has $\leq k$ equivalence classes.
Claim: If $\equiv_L$ has $k$ equivalence classes, then $L$ can be accepted by a DFA with $k$ states.

Let $L$ be a language s.t. $\equiv_L$ has $k$ equivalence classes, let them be $[x_1], [x_2], \ldots, [x_k]$. Then we'll construct a DFA $M = (Q, \Sigma, \delta, q_0, F)$ where:

- $Q = \{ [x_1], [x_2], \ldots, [x_k] \}$ (equiv. classes are the states)
- $q_0 = [x_0]$ (start state)
- $F = \{ [y] \mid y \in L \}$ (accept states)
- $\delta([x], a) = [x \sigma]$ for every equiv. class $[x] \in Q$ and every $a \in \Sigma$

$\square$
Thm [Myhill-Nerode]: $L$ is regular iff the number of equivalence classes of $\equiv_L$ is finite.

Using Myhill-Nerode to prove nonregularity:

$L_3 = \{ 0^k1^k \mid k > 0 \}$

Look 3 slides back: we have $\infty$ number of equiv. classes: $[\epsilon], [0], [00], [000], \ldots, [0^n], \ldots$

$L_4 = \{ ww^R \mid w \in \{0,1\}^* \}$

Game: want to show $\infty$ number of equiv. classes:

Want to show that every string has its own equiv. class

In particular, for any $x, y$ s.t. $x+y$
Claim: a DFA is minimal iff its number of states is the same as the number of equivalence classes of its language.
Suppose we have a DFA - how to construct a corresponding minimal DFA?

1) Get rid of unreachable states
   - e.g. do a BFS from the initial state
Suppose we have a DFA - how to construct a corresponding minimal DFA?

1. Remove unreachable states.
1. Remove unreachable states.
2. Identify equivalent states (and merge them):
   - construct graph with vertices = states
   - place edges between every accept and nonaccept state
1. Remove unreachable states.
2. Identify equivalent states (and merge them):
   - construct graph with vertices = states
   - place edges between every accept and nonaccept state
   - continue placing edges as follows while can:

   for $q, r \in Q$, $q \neq r$, place edge $(q, r)$
   if there exists $a \in \Sigma$ s.t.
   $(\delta(q, a), \delta(r, a))$ is an edge.
1. Remove unreachable states.
2. Identify equivalent states (and merge them):
   - construct graph with vertices = states
   - place edges between every accept and nonaccept state
   - continue placing edges as follows while can:
     for \( q, r \in Q \), \( q \neq r \), place edge \( (q, r) \)
     if there exists \( a \in \Sigma \) s.t.
     \( (\delta(q, a), \delta(r, a)) \) is an edge.
   - merge all states that do not have edges between them into a single state