MYHILL-NERODE Theorem

- Given a language $L$, any two strings $x$ and $y$ are in the same class if for all possible strings $z$ either both $xz$ and $yz$ are in $L$ or both not. These classes are equivalent classes.

1. $L$ divides the set of all possible strings into separate (mutually exclusive) classes.
2. If $L$ is regular, the number of classes $L$ creates is finite.
3. If the number of classes $L$ creates is finite, $L$ is regular.
MYHILL-NERODE Theorem: Example

- We may use MYHILL-NERODE theorem to prove a language is regular or is not regular.

- Example: $0^*10^*$
  three equivalent classes $c_1 = 0^*$, $c_2 = 0^*10^*$, and $c_3 = 0^*10^*1(0+1)^*$

- Example: $a^n b^n$
  it creates infinite number of classes: $\varepsilon$, $a$, $aa$, $aaa$, $\ldots,$
Minimum State automaton

- The minimum state automaton accepting a language $L$ is unique. The number of states of the minimum state automaton is the number equivalent classes created by $L$.

- Example: $0^*10^*$

  the minimum FA for this language has three states each for an equivalent class.
Minimization Algorithm

- For a given DFA, two states $p$ and $q$ are equivalent if and only if for each input string $x$, $\delta(p, x)$ is an accepting state if and only if $\delta(q, x)$ is an accepting state.

- Remove all states that are not accessible from the initial state. The result is the minimum state finite automaton.

- Find all equivalent states.

- Equivalent states are combined into one state.
Find Equivalent States

For p in F and q in Q-F do mark (p, q)
for each pair of distinct states (p, q) in F × F or (Q-F) ×(Q-F) do
    if for some input symbol a, (δ(p, a), δ(q, a)) is marked then
        mark (p, q);
        recursively mark all unmarked pairs on the list for (p, q) and on
        the lists of other pairs that are marked at this step.
    else
        for all input symbols a do
            if (δ(p, a) ≠ δ(q, a)
                put (p, q) on the list for (δ(p, a), δ(q, a))
        All unmarked pairs of states are equivalent states.
Equivalent States

- The find equivalent state algorithm finds all equivalent states.
- The equivalent states found by the algorithm are equivalent.
- Equivalence of states is transitive. Namely, if $p$ is equivalent to $q$ and $q$ is equivalent to $r$, then $p$ is equivalent to $r$.
- Equivalence of states partitions all states into disjoint sets.
Decidability

• A problem is decidable if there exists an algorithm to determine the answer of the problem within a given number of steps.

• A problem is undecidable if there is no such an algorithm.
Decidable Problems for Regular Languages

- If a regular language is empty
- If a regular language is finite
- If a regular language is infinite
- Equivalence of two regular languages
Emptiness, Finiteness, and Infiniteness

- The language accepted by an DFA $M$ with $n$ states is
  - nonempty if and only if the DFA accepts a string of length less than $n$.
  - infinite if and only if the DFA accepts some string of length $l$, where $n \leq l < 2n$.

There is an algorithm to check if all strings $w$ with $|w| < n$ belong to a regular language.

There is an algorithm to check if all strings $w$ with $n \leq |w| < 2n$ belong to a regular language.
Equivalence

- There is an algorithm to determine if two regular languages are equivalent.

- Assume that two languages are represented as regular expressions.

- $L_1$ and $L_2$ are equivalent if and only if $(\overline{L_1} \cap L_2) \cup (L_1 \cap \overline{L_2})$ is empty. $(\overline{L_1} \cap L_2) \cup (L_1 \cap \overline{L_2})$ is a regular language and there is an algorithm to determine its emptiness.
Equivalence

- Assume that two languages are represented as the minimum DFAs, M1 and M2.
- Imagine a new DFA M which includes all states and transitions of M1 and M2.
- Apply the find equivalent state algorithm to determine if the two start states of M1 and M2 are equivalent or not.
- If they are equivalent, then the two languages are equivalent. Otherwise, they are not equivalent.