Showing Languages are Regular or Not Regular

Two main techniques:

- **Pumping Lemma.** Used to show that a language is **not** regular.

- **Myhill-Nerode Theorem.**
  - Used to classify a language as regular or non-regular.
  - Also used to construct minimum state DFAs.
What is Pumping?

- For the regular expression \( r = (a \cup b)(abb)^*(bba \cup bbb) \), the string \( w = babbbbbb \) pumps because we may decompose \( w \) into three substrings \( w = b \cdot abb \cdot bbb \) such that \( b(abb)^ibbb \) is represented by \( r \), for every \( i \geq 0 \).

- In this DFA

![DFA Diagram]

the string \( aaababa = aa \cdot aba \cdot b \) pumps because of the cycle that the \( aba \) follows.
Pumping Lemma (Theorem 7.6.3).
Suppose that \( L \) is a regular language. Then there exists an integer \( k \) such that for all strings \( z \in L \) satisfying \( |z| \geq k \), we can write \( z = uvw \) where

1. \( |v| \geq 1 \);  

2. \( |uv| \leq k \); and  

3. \( uv^iw \in L \), for all \( i \geq 0 \).
Proof of the Pumping Lemma. Let $M$ be a DFA accepting $L$. Let $k$ be the number of states in $M$. Let $z$ be any string in $L$ satisfying $|z| \geq k$. Consider the accepting computation for $z$

\[(q_0, z) \xrightarrow{\ast} (q_f, \lambda),\]

where $q_f$ is a final state. This computation contains $|z| + 1 > k$ distinct configurations. Hence some state is repeated in these configurations (Pigeonhole Principle). Choose $q_j$ to be the state that is repeated the earliest and write that earliest repetition as follows:

\[(q_0, uvw) \xrightarrow{\ast} (q_j, vw) \xrightarrow{\dagger} (q_j, w) \xrightarrow{\ast} (q_f, \lambda).\]

Then (1) $|v| \geq 1$; (2) $|uv| \leq k$; and (3) $uv^i w \in L$, for all $i \geq 0$. 

**Proving a Language is Not Regular**

The logical structure of the Pumping Lemma is as follows:

\[ p \Rightarrow q \]

\[ L \text{ is regular } \Rightarrow \exists k \ \forall z \ (z \in L \land |z| \geq k) \Rightarrow \]

\[ \exists u, v, w \ (z = uvw \land \]

\[ |v| \geq 1 \land |uv| \leq k \land \]

\[ (\forall i \geq 0 \ uv^i w \in L) \].

Hence, showing \( \neg q \) suffices to show \( L \) is not regular. What is \( \neg q \)?

\[ \neg q = \forall k \ \exists z \ (z \in L \land |z| \geq k \land \]

\[ \forall u, v, w \ (z \neq uvw \lor \]

\[ |v| = 0 \lor |uv| > k \lor \]

\[ (\exists i \geq 0 \ uv^i w \notin L) \].
Proof Strategy

- Start with a language \( L \) that you want to show is not regular.

- Let \( k \) be an arbitrary positive integer.

- Choose \( z \in L \) such that \( |z| \geq k \) and \( z \) does not pump. (This is the creative part.) Note that \( z \) is a function of \( k \).

- Show that for any decomposition \( z = uvw \) with \( |v| \geq 1 \) and \( |uv| \leq k \), there exists an \( i \geq 0 \) such that \( uv^i w \not\in L \).

The argument is often presented as a proof by contradiction, but this is completely unnecessary.
EXAMPLE.

The language

\[ L_1 = \{a^n b^n \mid n \geq 0\} \]

is not regular.

**Proof:** Let \( k > 0 \) be arbitrary. Let \( z = a^k b^k \).
Then \( z \in L_1 \) and \( |z| \geq k \).

Let \( z = uvw \) be any decomposition of \( z \)
satisfying \( |v| \geq 1 \) and \( |uv| \leq k \). Then \( u = a^r \)
and \( v = a^s \), where \( 0 \leq r \) and \( 1 \leq s \leq k \). Then
\( uv^2w = a^{k+s} b^k \notin L_1 \).

As \( L_1 \) does not satisfy the conclusion of the
Pumping Lemma, it cannot be regular.
EXAMPLE.

The language of palindromes

\[ L_2 = \{ w \in \{0, 1\} | w^R = w \} \]

is not regular.

**Proof:** Let \( k > 0 \) be arbitrary. Let \( z = 0^k 10^k \).
Then \( z \in L_2 \) and \( |z| \geq k \).

Let \( z = uvw \) be any decomposition of \( z \)
satisfying \( |v| \geq 1 \) and \( |uv| \leq k \). Then \( u = 0^r \)
and \( v = 0^s \), where \( 0 \leq r \) and \( 1 \leq s \leq k \). Then
\( uv^2w = 0^{k+s}10^k \notin L_2 \).

As \( L_2 \) does not satisfy the conclusion of the
Pumping Lemma, it cannot be regular.
A Tricky Example

Show that the language

\[ L_3 = \{ w \in \{0, 1\}^* \mid n_0(w) \neq n_1(w) \} \]

is not regular.
Corollary 7.6.5. Let $M$ be a DFA. Then there is an algorithm to determine whether

1 $L(M)$ is empty.
2 $L(M)$ is finite.
3 $L(M)$ is infinite.

Corollary 7.6.6. Let $M_1$ and $M_2$ be DFAs. Then there is an algorithm to determine whether $L(M_1) = L(M_2)$.

Proof: Equivalent to determining whether

$$(L(M_1) \cap \overline{L(M_2)}) \cup (\overline{L(M_1)} \cap L(M_2)) = \emptyset.$$
Partitioning Reviewed

Start with a DFA $M = (Q, \Sigma, \delta, q_0, F)$.

Suppose that $Q = \{q_0, q_1, \ldots, q_{n-1}\}$.

For each state $q_i$, define the associated language

$$L_{q_i} = \{w \in \Sigma^* \mid \tilde{\delta}(q_0, w) = q_i\}.$$  

Then

$$L_{q_0}, L_{q_1}, \ldots, L_{q_{n-1}}$$

is a partition of $\Sigma^*$. 
Example of Partitioning

Consider the DFA
\[ M_1 = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta_1, q_0, \{q_2\}) \] :

The partition generated by \( M_1 \) is
\[
L_{q_0} = b^* \\
L_{q_1} = b^*ab^* \\
L_{q_2} = b^*ab^*ab^* \\
L_{q_3} = b^*ab^*ab^*a(a \cup b)^*
\]

The corresponding equivalence relation on \((a \cup b)^*\) is as follows:

For two strings \( u, v \in (a \cup b)^* \), we have
\( u \equiv_{M_1} v \) if and only if \( \tilde{\delta}_1(q_0, u) = \tilde{\delta}_1(q_0, v) \).
Example of Partitioning
(Continued)

The language accepted by $M_1$ is

$$L_1 = L_{q_2} = b^*ab^*ab^*.$$
Right-Invariance

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Strings $u, v \in \Sigma^*$ are indistinguishable by $M$ if $\tilde{\delta}(q_0, u) = \tilde{\delta}(q_0, v)$. This is written $u \equiv_M v$.

**Lemma 1.** The relation $\equiv_M$ is an equivalence relation.

An equivalence relation $\equiv$ on $\Sigma^*$ is right-invariant if $u \equiv v$ implies $uw \equiv vw$ for all $w \in \Sigma^*$.

**Lemma 2.** The relation $\equiv_M$ is right-invariant.
Another Equivalence Relation

Let \( L \subseteq \Sigma^* \) be a language. Strings \( u, v \in \Sigma^* \) are indistinguishable by \( L \) if, for every \( w \in \Sigma^* \), either \( uw, vw \in L \) or \( uw, vw \notin L \). This is written \( u \equiv_L v \).

**EXAMPLE.** Let

\[
L_2 = \{a^i b^i \mid 0 \leq i\}.
\]

What are the equivalence classes of \( \equiv_{L_2} \)?

\[\square\]
Another Equivalence Relation

**Lemma 3.** The relation $\equiv_L$ is an equivalence relation.

**Lemma 4.** Let $M$ be a DFA. Then every equivalence class of $\equiv_L(M)$ is a union of equivalence classes of $\equiv_M$. 
Example

Another DFA $M'_1$ that accepts $L_1$ is given by

\[ L_{p_0} = (bb)^* \]
\[ L_{p_1} = (bb)^* a \]
\[ L_{p_2} = ((bb)^* a a \cup L_{p_6} b) b^* \]
\[ L_{p_3} = (L_{p_2} \cup L_{p_6}) a (a \cup b)^* \]
\[ L_{p_4} = b(bb)^* \]
\[ L_{p_5} = (b(bb)^* a \cup L_{p_1} b) b^* \]
\[ L_{p_6} = (b(bb)^* a (bb)^* ab) b^* a \]

Express the equivalence classes of $\equiv_{L_1}$ using the partition ?
**Myhill-Nerode Theorem.** The following are equivalent for a language $L$ over $\Sigma$:

(i) $L$ is regular.

(ii) There is a right-invariant equivalence relation $\equiv$ on $\Sigma^*$ with a finite number of equivalence classes such that $L$ is the union of some of the equivalence classes of $\equiv$.

(iii) $\equiv_L$ has a finite number of equivalence classes.
Proof:

(i) $\Rightarrow$ (ii). Choose a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts $L$. Show that $\equiv_M$ satisfies (ii).

(ii) $\Rightarrow$ (iii). Let $\equiv$ satisfy (ii). Show that $[u]_{\equiv} \subseteq [u]_{\equiv_L}$, for every string $u \in \Sigma^*$. Show that (iii) holds.

(iii) $\Rightarrow$ (i). Build a DFA $M_L$ from $\equiv_L$. 
Examples

Assume $\Sigma = \{0, 1\}$.

$L_3 = \{0^{2^i} | 0 \leq i \}$

$= \{0, 00, 0000, 00000000, \ldots \}$

Equivalence classes of $\equiv_{L_3}$ ?

Apply Myhill-Nerode ?

$L_4 = \{0^{2^i} | 0 \leq i \}$

$= \{\lambda, 00, 0000, 000000, \ldots \}$

Equivalence classes of $\equiv_{L_4}$ ?

What is the corresponding DFA ?
Minimum-State DFA

Theorem 7.7.5. The DFA $M_L$ constructed in the Myhill-Nerode Theorem is the unique (up to relabeling states) minimum-state DFA that accepts $L$. 
Minimization Algorithm

DFA-Minimization($M$)

▷ $M = (Q, \Sigma, \delta, q_0, F)$ is a DFA.
▷ Assume that $F \neq Q$ and $F \neq \emptyset$.
▷ $P$ is a partition of $Q$.
▷ $\equiv_P$ is the corresponding equivalence relation.

$P \leftarrow \{F, Q - F\}$

repeat done $\leftarrow$ true
  for $S \in P$
    do for $q_i, q_j \in S, q_i \neq q_j$
      do for $a \in \Sigma$
        do if $\delta(q_i, a) \neq_P \delta(q_j, a)$
          then done $\leftarrow$ false
          refine $S$ based on $a$
  until done

return DFA constructed from $P$
Example*

Apply the state minimization algorithm to this DFA:

* Hopcroft and Ullman, page 68.