Properties of Regular Languages

Reading: Chapter 4
Pumping Lemma for Regular Languages

- **Lemma**: (the pumping lemma)
  Let $M$ be a DFA with $|Q| = n$ states, and let $x$ be a string in $L(M)$, where $|x| \geq n$. Then $x = uvw$, where $u,v,$ and $w$ are all in $\Sigma^*$ and:
  - $1 \leq |uv| \leq n$
  - $|v| \geq 1$
  - $u^i v^j w$ is in $L(M)$, for all $i \geq 0$

- Note this statement of the pumping lemma is slightly different than the books’.
Pumping Lemma for Regular Languages

• Proof:
  Let $M$ be a DFA where $|Q| = n$, and let $x$ be a string in $L(M)$ where $|x| \geq n$. Furthermore, let $x = a_1a_2 \ldots a_m$ where $\delta(q_0, a_1a_2 \ldots a_p) = q_{jp}$.

  \[
  \begin{array}{cccccc}
  a_1 & a_2 & a_3 & \ldots & a_m \\
  q_{j0} & q_{j1} & q_{j2} & q_{j3} & \ldots & q_{jm}
  \end{array}
  \]
  \[
  m \geq n \text{ and } q_{j0} \text{ is } q_0
  \]

  Consider the first $n$ symbols, and first $n+1$ states on the above path:

  \[
  \begin{array}{cccccc}
  a_1 & a_2 & a_3 & \ldots & a_n \\
  q_{j0} & q_{j1} & q_{j2} & q_{j3} & \ldots & q_{jn}
  \end{array}
  \]

  Since $|Q| = n$, it follows from the pigeon-hole principle that $j_s = j_t$ for some $0 \leq s < t \leq n$, i.e., some state appears on this path twice (perhaps many states appear more than once, but at least one does).
The diagram illustrates a sequence of states and transitions between them. The states are labeled as follows:

- $q_0$, $q_{j1}$, $q_{js}$, $q_{jt}$, $q_{jn}$

The transitions are labeled with symbols $a_1, a_2, \ldots, a_s, a_{s+1}, a_t, a_{t+1}, a_n$.

The key points are:

- The sequence $a_1 \ldots a_s$ leads to $q_{js} = q_{jt}$.
- The sequence $a_{s+1} \ldots a_t$ is a cycle back to $q_{jn}$.

The diagram also includes arrows indicating the same states and the cycle.
• Let:
  
  – \( u = a_1 \ldots a_s \)
  
  – \( v = a_{s+1} \ldots a_t \)

• Since \( 0 <= s < t <= n \) and \( uv = a_1 \ldots a_t \) it follows that:
  
  – \( 1 <= |v| \) and therefore \( 1 <= |uv| \)
  
  – \( |uv| <= n \) and therefore \( 1 <= |uv| <= n \)

• In addition, let:
  
  – \( w = a_{t+1} \ldots a_m \)

• It follows that \( uv^iw = a_1 \ldots a_s (a_{s+1} \ldots a_t)^ia_{t+1} \ldots a_m \) is in \( L(M) \), for all \( i >= 0 \).

  In other words, when processing the accepted string \( x \), the cycle was traversed once, but could have been traversed as many times as desired, and the resulting string would still be accepted.
Example:

\[ n = 5 \]

\[ x = 0001000 \text{ is in } L(M) \]

\[ u = 0 \]
\[ v = 001 \]
\[ w = 000 \quad \text{uv}^i w \text{ is in } L(M), \text{ i.e., } 0(001)^i000 \text{ is in } L(M), \text{ for all } i \geq 0 \]
• Note this does not mean that every string accepted by the DFA has this form:
  – 001 is in $L(M)$ but is not of the form $0(001)^i000$

• Similarly, this doesn’t even mean that every long string accepted by the DFA has this form:
  – 0011111 is in $L(M)$, is very long, but is not of the form $0(001)^i000$

• Note, however, in this latter case 0011111 could be similarly decomposed.
• **Note:** It may be the case that no $x$ in $L(M)$ has $|x| \geq n$. 
• Example:

\begin{center}
\begin{tikzpicture}
  \node[state, initial] (q0) at (0,0) {$q_0$};
  \node[state, right of=q0] (q1) {$q_1$};
  \node[state, below of=q1] (q3) {$q_3$};
  \node[state, left of=q1] (q2) {$q_2$};
  \draw[-stealth] (q0) edge [loop above] node {b} ();
  \draw[-stealth] (q0) edge node {a} (q1);
  \draw[-stealth] (q0) edge node {a} (q2);
  \draw[-stealth] (q1) edge [loop above] node {a} ();
  \draw[-stealth] (q1) edge node {b} (q2);
  \draw[-stealth] (q1) edge node {b} (q3);
  \draw[-stealth] (q2) edge node {a,b} (q3);
\end{tikzpicture}
\end{center}

\[ n = 4 \]

\[ x = bbbab \text{ is in } L(M) \]

\[ |x| = 5 \]

\[ u = \varepsilon \]

\[ v = b \]

\[ w = bbab \]

\[ (b)^i bbab \text{ is in } L(M), \text{ for all } i \geq 0 \]
Example:

\[ x = bbbab \text{ is in } L(M) \]
\[ |x| = 5 \]

\[ u = b \]
\[ v = b \]
\[ w = bab \]

\[ b(b)^i bab \text{ is in } L(M), \text{ for all } i \geq 0 \]
• Example:

\[
\begin{array}{c}
\text{q}_0 \xrightarrow{a} \text{q}_1 \xrightarrow{b} \text{q}_0 \xrightarrow{b} \text{q}_1 \xrightarrow{a} \text{q}_0 \\
\text{q}_2 \xrightarrow{a,b} \text{q}_3 \\
\end{array}
\]

\[n = 4\]

\[
x = \text{bbbab} \text{ is in } L(M)
\]

|\[|x| = 5\]|

\[
u = \text{bb}
\]

\[
v = \text{b}
\]

\[
w = \text{ab}
\]

\[
\text{bb(b)}^i \text{ab} \text{ is in } L(M), \text{ for all } i \geq 0
\]
• Example:

\[ x = \text{bbbab is in } L(M) \]
\[ |x| = 5 \]

\[ u = b \]
\[ v = bb \]
\[ w = ab \]

\[ b(bb)^iab \text{ is in } L(M), \text{ for all } i \geq 0 \]
NonRegularity Example

• **Theorem:** The language:

\[ L = \{0^k1^k \mid k \geq 0\} \]  \hspace{1cm} (1)

is not regular.

• **Proof:** (by contradiction) Suppose that \( L \) is regular. Then there exists a DFA \( M \) such that:

\[ L = L(M) \]  \hspace{1cm} (2)

We will show that \( M \) accepts some strings not in \( L \), contradicting (2).

Suppose that \( M \) has \( n \) states, and consider a string \( x = 0^m1^m \), where \( m >> n \).

By (1), \( x \) is in \( L \).

By (2), \( x \) is also in \( L(M) \).
Since $x$ is very long, i.e., $|x| = 2^m >> n$, it follows from the pumping lemma that:

- $x = uvw$
- $1 \leq |uv| \leq n$
- $1 \leq |v|$, and
- $uv^i w$ is in $L(M)$, for all $i \geq 0$

Since $1 \leq |uv| \leq n$ and $n << m$, it follows that $1 \leq |uv| \leq m$.

Also, since $x = 0^m 1^m$ it follows that $uv$ is a substring of $0^m$.

In other words $v = 0^j$, for some $j \geq 1$.

Since $uv^i w$ is in $L(M)$, for all $i \geq 0$, it follows that $0^{m+cj} 1^m$ is in $L(M)$, for all $c \geq 1$.

But by (2), $0^{m+cj} 1^m$ is in $L$, for any $c \geq 1$, a contradiction. •
NonRegularity Example

• **Theorem:** The language:
  \[ L = \{0^k1^k2^k \mid k \geq 0\} \]  
  is not regular.

• **Proof:** (by contradiction) Suppose that L is regular. Then there exists a DFA M such that:
  \[ L = L(M) \]  

We will show that M accepts some strings not in L, contradicting (2).

Suppose that M has n states, and consider a string \( x = 0^m1^m2^m \), where \( m \gg n \).

By (1), \( x \) is in L.

By (2), \( x \) is also in \( L(M) \).
Since is very long, i.e., $|x| = 3^*m >> n$, it follows from the pumping lemma that:

- $x = uvw$
- $1 \leq |uv| \leq n$
- $1 \leq |v|$, and
- $uv^i w$ is in $L(M)$, for all $i \geq 0$

Since $1 \leq |uv| \leq n$ and $n << m$, it follows that $1 \leq |uv| \leq m$.

Also, since $x = 0^m1^m2^m$ it follows that $uv$ is a substring of $0^m$.

In other words $v = 0^j$, for some $j \geq 1$.

Since $uv^i w$ is in $L(M)$, for all $i \geq 0$, it follows that $0^{m+cj}1^m2^m$ is in $L(M)$, for all $c \geq 1$.

But by (2), $0^{m+cj}1^m2^m$ is in $L$, for any $c \geq 1$, a contradiction.
Theorem: The language:
\[ L = \{0^m1^n2^{m+n} \mid m,n \geq 0\} \] (1)
is not regular.

Proof: (by contradiction) Suppose that L is regular. Then there exists a DFA M such that:
\[ L = L(M) \] (2)

We will show that M accepts some strings not in L, contradicting (2).

Suppose that M has n states, and consider a string \( x = 0^m1^n2^{m+n} \), where \( m \gg n \).

By (1), x is in L.

By (2), x is also in L(M).
Since $|x| = m >> n$, it follows from the pumping lemma that:

- $x = uvw$
- $1 <= |uv| <= n$
- $1 <= |v|$, and
- $uv^i w$ is in $L(M)$, for all $i >= 0$

Since $1 <= |uv| <= n$ and $n << m$, it follows that $1 <= |uv| <= m$.

Also, since $x = 0^m 1^n 2^{m+n}$ it follows that $uv$ is a substring of $0^m$.

In other words $v = 0^j$, for some $j >= 1$.

Since $uv^i w$ is in $L(M)$, for all $i >= 0$, it follows that $0^{m+cj} 1^m 2^{m+n}$ is in $L(M)$, for all $c >= 1$. In other words $v$ can be “pumped” as many times as we like, and we still get a string in $L(M)$.

But by (2), $0^{m+cj} 1^n 2^{m+n}$ is in $L$, for any $c >= 1$, a contradiction.

- Note that the above proof is almost identical to the previous proof.
• By the way…

\{0^n1^n \mid 0\leq n\} \quad \text{Is not regular}

\{0^n1^n \mid 0\leq n\leq k, \text{for some fixed } k\} \quad \text{Is regular, for any fixed } k.

• For \(k=3\):

\[ L = \{\varepsilon, 01, 0011, 000111\} \]
The following theorems will all exploit the following observations…

Let $\Sigma$ be an alphabet, and let $|\Sigma| = k$.

How many strings of length $n$ are there?

$$k^n$$

For example, if $\Sigma = \{0,1\}$ and $n = 3$, then there are 8 strings of length 3.
• **Theorem:** Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Then $L(M)$ is non-empty iff there exists an $x$ in $L(M)$ such that $|x| < |Q|$. 

• **Proof:**

(if) Suppose there exists a string $x$ in $L(M)$ such that $|x| < |Q|$. Then clearly $L(M)$ is non-empty.
• **Theorem:** Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Then $L(M)$ is non-empty iff there exists an $x$ in $L(M)$ such that $|x| < |Q|$. 

(only if) By contradiction.

Suppose that $L(M)$ is non-empty, but that there exists no string $x$ in $L(M)$ such that $|x| < |Q|$. It follows that $|y| \geq |Q|$, for all $y$ in $L(M)$.

Let $z$ be a string of shortest length in $L(M)$. Then $|z| \geq |Q|$. By the pumping lemma $z=uvw$, $|v|\geq 1$ and $uv^iw$ is in $L(M)$ for all $i\geq 0$.

But then $uv^0w = uw$ is in $L(M)$ and:

$$|uw| = |z| - |v|$$
$$\leq |z| - 1 \quad \text{because } |v| \geq 1$$
$$< |z|$$

Since $uw$ is in $L(M)$, it follows that $z$ is not a string of shortest length in $L(M)$, a contradiction.
• **Corollary:** Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Then there is an algorithm to determine if $L(M)$ is empty.

• **Proof:**
From the theorem it follows that if $L(M)$ is non-empty then there exists a string $x$ where $|x| < n$ and $n = |Q|$ such that $M$ accepts $x$. We can try running $M$ on each string of length $< n$ to see if any are accepted. If one is accepted then $L(M)$ is non-empty, otherwise $L(M)$ is empty. Since the number of states $|Q|$ and the number of input symbols $|\Sigma|$ are both fixed, it follows that there are at most a finite number of strings (of length less than $|Q|$) that need to be tested.

• Note that a simple graph search algorithm works here too…in fact, the above algorithm is a graph search…
• **Theorem:** Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Then $L(M)$ is finite iff $|x| < |Q|$ for all $x$ in $L(M)$.

• **Proof:**

(if) Suppose that $|x| < |Q|$ for all $x$ in $L(M)$. Since the number of states $|Q|$ and the number of input symbols $|\Sigma|$ are both fixed, it follows that there are at most a finite number of strings of length less than $|Q|$. It follows that $L(M)$ is finite (exercise: give an upper bound on the number of such strings).

(only if) By contradiction. Suppose that $L(M)$ is finite, but that $|x| \geq |Q|$ for some $x$ in $L(M)$. From the pumping lemma it follows that $x=uvw$, $|v|\geq 1$ and $uv^iw$ is in $L(M)$ for all $i\geq 0$. But then $L(M)$ would be infinite, a contradiction. •
• **Theorem:** Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Then $L(M)$ is infinite iff there exists an $x$ in $L(M)$ such that $|x| \geq |Q|$.

• **Proof:**

(if) Suppose there exists an $x$ in $L(M)$ such that $|x| \geq |Q|$. From the pumping lemma it follows that $x=uvw$, $|v|\geq1$ and $uv^iw$ is in $L(M)$ for all $i\geq0$. Therefore $L(M)$ is infinite.

(only if) By contradiction. Suppose that $L(M)$ is infinite, but that there is no $x$ in $L(M)$ such that $|x| \geq |Q|$. It follows that each $x$ in $L(M)$ has length less than $|Q|$. Since the number of states $|Q|$ and the number of input symbols $|\Sigma|$ are both fixed, it follows that there are at most a finite number of strings of length less than $|Q|$. It follows that $L(M)$ is finite. A contradiction.

• Note that the above also follows directly from the previous theorem.
• **Theorem:** Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and let $n = |Q|$. Then $L(M)$ is infinite iff there exists an $x$ in $L(M)$ such that $n \leq |x| < 2n$.

• **Proof:** (left as an exercise; similar to the previous theorem).

• **Corollary:** Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Then there is an algorithm to determine if $L(M)$ is infinite.

• **Proof:** (left as an exercise).
Closure Properties of Regular Languages

• Consider various operations on languages:

\[ L \] = \{x \mid x \text{ is in } \Sigma^* \text{ and } x \text{ is not in } L \}
\[ = \Sigma^* - L \]

\[ L_1 \cup L_2 \] = \{x \mid x \text{ is in } L_1 \text{ or } L_2 \}

\[ L_1 \cap L_2 \] = \{x \mid x \text{ is in } L_1 \text{ and } L_2 \}

\[ L_1 - L_2 \] = \{x \mid x \text{ is in } L_1 \text{ but not in } L_2 \}

\[ L_1 L_2 \] = \{xy \mid x \text{ is in } L_1 \text{ and } y \text{ is in } L_2 \}

\[ L^* \] = \bigcup_{i=0}^{\infty} L^i = L^0 U L^1 U L^2 U...

\[ L^+ \] = \bigcup_{i=1}^{\infty} L^i = L^1 U L^2 U...

• A closure property for the regular languages states that if a set of one or more languages is regular, then some operation on that set results in a regular language.
• **Theorem:** Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Then $M' = (Q, \Sigma, \delta, q_0, Q-F)$ is a DFA where $L(M') = \Sigma^* - L(M)$.

• **Proof:** (left as an exercise).

• **Corollary:** The regular languages are closed under complementation, i.e., if $L$ is a regular language then so is $\overline{L}$.

• Does the above construction work for NFAs?
• **Theorem:** The regular languages are closed with respect to union, concatenation and Kleene closure.

• **Proof:** Let $L_1$ and $L_2$ be regular languages. Then there exist regular expressions $r_1$ and $r_2$ such that $L(r_1) = L_1$ and $L(r_2) = L_2$. But then $r_1 + r_2$ is a regular expression representing $L_1 \cup L_2$. Similarly for concatenation and Kleene closure.
How about Intersection?

- **Lemma:** Let $L_1$ and $L_2$ be subsets of $\Sigma^*$. Then $L_1 \cup L_2 = \overline{L_1} \cap \overline{L_2}$
• **Theorem:** Let $L_1$ and $L_2$ be regular languages. Then $L_3 = L_1 \cap L_2$ is a regular language.

• **Proof:** (algebraic proof) Let $L_1$ and $L_2$ be regular languages. Then:

- $\overline{L}_1$ is regular, because the regular languages are closed wrt complementation
- $\overline{L}_2$ is regular, because the regular languages are closed wrt complementation
- $\overline{L}_1 \cup \overline{L}_2$ is regular, because the regular languages are closed wrt union
- $\overline{L}_1 \cup \overline{L}_2$ is regular, because the regular languages are closed wrt complementation

But by the lemma:

$$\overline{L}_1 \cup \overline{L}_2 = \overline{L}_1 \cap \overline{L}_2 = L_1 \cap L_2 \quad \bullet$$
• A direct construction of a DFA $M$ such that $L(M) = L_1 \cap L_2$:

• Let:
  
  $M_1 = (Q_1, \Sigma, \delta_1, p_0, F_1)$, where $Q_1 = \{p_0, p_1, \ldots\}$
  
  $M_2 = (Q_2, \Sigma, \delta_2, q_0, F_2)$, where $Q_2 = \{q_0, q_1, \ldots\}$

  where $L(M_1) = L_1$ and $L(M_2) = L_2$.

• Construct $M$ where:
  
  $Q = Q_1 \times Q_2$
  
  $\Sigma = \text{as with } M_1 \text{ and } M_2$

  $F = F_1 \times F_2$

  start state = $[p_0, q_0]$

  $\delta([p_i, q_j], a) = [\delta_1(p_i, a), \delta_2(q_j, a)]$ for all $[p_i, q_j]$ in $Q$ and $a$ in $\Sigma$
Example:

The construct gives:

\[ Q = Q_1 \times Q_2 = \{[p_0, q_0], [p_0, q_1], [p_0, q_2], [p_1, q_0], [p_1, q_1], [p_1, q_2]\} \]

\[ \Sigma = \{0, 1\} \]

\[ F = \{[p_1, q_2]\} \]

start state = \([p_0, q_0]\)

\[ \delta - \text{Some examples:} \]
\[ \delta([p_0, q_2], 0) = [\delta_1(p_0, 0), \delta_2(q_2, 0)] = [p_1, q_2] \]
\[ \delta([p_1, q_2], 1) = [\delta_1(p_1, 1), \delta_2(q_2, 1)] = [p_1, q_2] \]
• Final Result:

- Question: What if $\Sigma_1$ does not equal $\Sigma_2$?

- Try all three machines on: (what happens?)
  - 0100
  - 0101
  - 0011
  - 1010