Regular Languages

If we can define a language by RE, then it’s a **regular language**

**Theorem**

*If* $L_1$ *and* $L_2$ *are regular languages, then* $L_1 + L_2$ *(union)*, $L_1 L_2$ *(concatenation)*, and $L_1^*$ *(closure)* *are also regular languages.*

**Proof by Regular Expression.**

1. There exists REs $r_1$ and $r_2$ that define the regular languages $L_1$ and $L_2$
2. There exists an RE $(r_1 + r_2)$ that defines the language $L_1 + L_2$
3. There exists an RE $r_1 r_2$ that defines the language $L_1 L_2$
4. There exists an RE $r_1^*$ that defines the language $L_1^*$
5. All three of these sets of words are definable by RE

The set of regular languages is *closed* under union, concatenation, and Kleene closure.
Proof by Machines

1. Let us assume $TG_1$ and $TG_2$ exist that define languages $L_1$ and $L_2$ where each TG has a unique start and final state.

2. $L_1 + L_2$ can be described by:

3. $L_1L_2$ can be described by:

4. $L_1^*$ can be described by:
Proof by Machines

1. Let us assume $TG_1$ and $TG_2$ exist that define languages $L_1$ and $L_2$ where each TG has a unique start and final state.

2. $L_1 + L_2$ can be described by:

3. $L_1L_2$ can be described by:

4. $L_1^*$ can be described by:

Small problem for $L_1^*$ when the start has incoming edges. We must replicate the start state. We could convert to FA-$\lambda$ then to FA.
Example

\[ \Sigma = \{a \ b\} \]

\[ L_1 = \text{all words of 2+ letters that begin and end with the same letter} \]

\[ L_2 = \text{all words that contain the substring} aba \]

\[ r_1 = a(a + b)^*a + b(a + b)^*b \]

\[ r_2 = (a + b)^*aba(a + b)^* \]

\[ r_1 + r_2 = \]

\[ r_1r_2 = \]

\[ r_1^* = \]
Example

$\Sigma = \{a, b\}$

$L_1 = \text{all words of 2+ letters that begin and end with the same letter}$

$L_2 = \text{all words that contain the substring } aba$

$r_1 = a(a + b)^*a + b(a + b)^*b$

$r_2 = (a + b)^*aba(a + b)^*$

$r_1 + r_2 = [a(a + b)^*a + b(a + b)^*b] + [(a + b)^*aba(a + b)^*]$

$r_1r_2 =$

$r_1^* =$
Example

\[ \Sigma = \{a, b\} \]

\( L_1 = \) all words of 2+ letters that begin and end with the same letter

\( L_2 = \) all words that contain the substring \( aba \)

\( r_1 = a(a + b)^* a + b(a + b)^* b \)

\( r_2 = (a + b)^* aba(a + b)^* \)

\[ r_1 + r_2 = [a(a + b)^* a + b(a + b)^* b] + [(a + b)^* aba(a + b)^*] \]

\[ r_1 r_2 = [a(a + b)^* a + b(a + b)^* b] [(a + b)^* aba(a + b)^*] \]

\( r_1^* = \)
Example

\[ \Sigma = \{ a, b \} \]

\[ L_1 = \text{all words of 2+ letters that begin and end with the same letter} \]

\[ L_2 = \text{all words that contain the substring} aba \]

\[ r_1 = a(a + b)^*a + b(a + b)^*b \]

\[ r_2 = (a + b)^*aba(a + b)^* \]

\[ r_1 + r_2 = [a(a + b)^*a + b(a + b)^*b] + [(a + b)^*aba(a + b)^*] \]

\[ r_1r_2 = [a(a + b)^*a + b(a + b)^*b][(a + b)^*aba(a + b)^*] \]

\[ r_1^* = [a(a + b)^*a + b(a + b)^*b]^* \]
Example

Σ = \{a, b\}

L_1 = all words of 2+ letters that begin and end with the same letter

L_2 = all words that contain the substring aba

r_1 = a(a + b)^*a + b(a + b)^*b

r_2 = (a + b)^*aba(a + b)^*

r_1 + r_2 = [a(a + b)^*a + b(a + b)^*b] + [(a + b)^*aba(a + b)^*]

r_1 r_2 = [a(a + b)^*a + b(a + b)^*b] [(a + b)^*aba(a + b)^*]

r_1^* = [a(a + b)^*a + b(a + b)^*b]^*

Show the TGs that accept L_1 and L_2
Show \(T G_1 + T G_2\), \(T G_1 T G_2\), and \(T G_1^*\)
Complements and Intersections

Definition

If \( L \) is a language over alphabet \( \Sigma \), we define its complement, \( L' \) to be the language of all strings of letters from \( \Sigma \) that are not words in \( L \).

Example

If \( L \) is the language over the alphabet \( \Sigma = \{a, b\} \) of all words that have a double \( a \) in them, then \( L' \) is the language of all words that do not have a double \( a \).

We must specify the alphabet \( \Sigma \) or else the complement of \( L \) might contain \( \text{cat, dog, } \ldots \) (because they are definitely not strings in \( L \)).

\[(L')' = L\]

for obvious reasons (theorem in set theory)
Complements and Regular Languages

Theorem
If $L$ is a regular language, then $L'$ is also a regular language. In other words, the set of regular languages is closed under complementation.

Proof.
- If $L$ is a regular language, we know from Kleene’s theorem that there is some FA that accepts $L$.
- The states of FA are each either final or non-final.
- Let us reverse the final status of each state (e.g. final $\rightarrow$ non-final, non-final $\rightarrow$ final).
- This new machine accepts all input strings the original FA rejected ($L'$). Likewise, the new machine rejects all input strings the original FA accepted ($L$).
- This new FA can be converted to an RE via Kleene’s theorem $\square$.
Complements of Regular Languages Example
Complements of Regular Languages Example

\[ q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_4 \]

Transitions:
- \( q_0 \rightarrow q_1 \) with label \( a \)
- \( q_1 \rightarrow q_2 \) with label \( b \)
- \( q_2 \rightarrow q_3 \) with label \( a, b \)
- \( q_3 \rightarrow q_4 \) with label \( a, b \)
- \( q_4 \rightarrow q_4 \) with label \( a, b \)
Language Intersection

Theorem

If $L_1$ and $L_2$ are regular languages, than $L_1 \cup L_2$ is also a regular language. e.g. the set of regular languages is closed under intersection.
Language Intersection

From the above, it is obvious how \((L_1' + L_2')' = L_1 \cup L_2\)
Algorithm for finding RE accepting $L_1 + L_2$

**Algorithm**

1. Define $r_1$ and $r_2$ which represent $L_1$ and $L_2$
2. Convert $r_1$ and $r_2$ to $FA_1$ and $FA_2$
3. Invert the states of $FA_1$ and $FA_2$ resulting in $FA_1'$ and $FA_2'$
4. Merge $FA_1'$ and $FA_2'$ into $TG'$, then convert $TG'$ into $FA_3'$
5. Invert the states of $FA_3'$, resulting in $FA_3$ (which accepts $L_1 \cup L_2$)

**Proof.**
Algorithm for finding RE accepting $L_1 + L_2$

**Algorithm**

1. Define $r_1$ and $r_2$ which represent $L_1$ and $L_2$
2. Convert $r_1$ and $r_2$ to $FA_1$ and $FA_2$
3. Invert the states of $FA_1$ and $FA_2$ resulting in $FA'_1$ and $FA'_2$
4. Merge $FA'_1$ and $FA'_2$ into $TG'$, then convert $TG'$ into $FA'_3$
5. Invert the states of $FA'_3$, resulting in $FA_3$ (which accepts $L_1 \cup L_2$)

**Proof.**

1. For a regular language, there exists a RE
2. Given an RE, there exists an FA (Kleene’s theorem)
3. We can complement an FA by swapping its states
4. We can describe $L'_1 + L'_2$ by merging two TGs
5. We can convert a TG to an RE
Example

$L_1 = $ all strings with a double $a$

$L_2 = $ all strings with an even number of $a$’s
Example

$L_1 =$ all strings with a double $a$
$L_2 =$ all strings with an even number of $a$’s

We can define $L_1$ and $L_2$ by the following REs:

$$r_1 = (a + b)^* aa (a + b)^*$$
$$r_2 = b^* (ab^* ab^*)^*$$
Example

$L_1 = \text{all strings with a double } a$

$L_2 = \text{all strings with an even number of } a\text{'s}$

We can define $L_1$ and $L_2$ by the following REs:

$r_1 = (a + b)^*aa(a + b)^*$

$r_2 = b^*(ab^*ab^*)^*$

Or the following FAs:
Example

Swapping the states:

Merging (Creating the TG):
Example

After converting the TG to FA:
Example

After swapping all of the states:

And converting the FA to RE with the bypass algorithm:

\[(a + abb*ab)^*a(a + bb*aab*a)(a + ab*a)^*\]
A Better Way...

- Remember creating a machine that accepts $FA_1 + FA_2$ where $FA_1$ has $x$-states, $FA_2$ has $y$-states, and our new machine has $z$-states
- We identify all final $z$-states by $x$-or-$y$ states being accepted upon the construction of our new machine
- Let’s change the designation for $FA_1 \cup FA_2$ to: All final $z$-states by $x$-and-$y$ states being accepted upon the construction of our new machine
- Now the new FA accepts only strings that reach simultaneously on both machines

**TL;DR** – change the rules of determining a final state of two FAs to be the intersection ($\cup$) rather than union ($+$)
One Final Example

Our two languages will be:

\[ L_1 = \text{all words that begin with an}a \]
\[ L_2 = \text{all words than end with an}a \]
\[ r_1 = a(a + b)^* \]
\[ r_2 = (a + b)^*a \]

An obvious solution is:

\[ a(a + b)^*a + a \]

But now we need to prove it...
Homework 6a

For each of the following pairs of regular languages, find a RE and FA that define $L_1 \cup L_2$

1. $(a + b)^*a \quad b(a + b)^*$
2. Even-length strings $ (b + ab)^*(a + \lambda) $  
3. Odd-length strings $ a(a + b)^*$
4. Even-length strings Strings with an even number of a’s